

А. Е. ФАТЕЕВ, А. И. РОЙТМАН, Т. П. ФАТЕЕВА

**ПРИКЛАДНЫЕ
ПРОГРАММЫ
В СИСТЕМЕ МАТЕМАТИЧЕСКОГО
ОБЕСПЕЧЕНИЯ
ЕС ЭВМ**

А. Е. ФАТЕЕВ, А. И. РОЙТМАН, Т. П. ФАТЕЕВА

**ПРИКЛАДНЫЕ
ПРОГРАММЫ
В СИСТЕМЕ МАТЕМАТИЧЕСКОГО
ОБЕСПЕЧЕНИЯ
ЕС ЭВМ**



**МОСКВА
„СТАТИСТИКА“ 1978**

- Ф27 Фатеев А. Е. и др. Прикладные программы в системе математического обеспечения ЕС ЭВМ. М., «Статистика», 1976.
184 с. с ил.

Перед загл. авт.: А. Е. Фатеев, А. И. Ройтман, Т. П. Фатеева.

В книге излагаются вопросы разработки, внедрения и эксплуатации пакетов прикладных программ в связи с особенностями технических средств и системного математического обеспечения ЕС ЭВМ. Реализация прикладных программ иллюстрируется на примере пакета прикладных программ СПУ/ЕС.

Книга предназначена для пользователей, применяющих в своей работе современные вычислительные средства. Кроме того, она будет полезна студентам, специализирующимся на разработке и внедрении АСУ.

30502-087
Ф 008(01)-76 67-76

6Ф7.3

ВВЕДЕНИЕ

В настоящее время в связи с организацией серийного производства ЕС ЭВМ и широким внедрением этих машин в народное хозяйство появилась необходимость в ознакомлении с их возможностями широкого круга специалистов — пользователей вычислительных машин. При этом пользователем мы называем любого специалиста, получающего результаты решения прикладных задач на ЭВМ и использующего их в своей практической деятельности. Можно утверждать, что дальнейшее распространение и эффективная эксплуатация ЭВМ третьего и последующих поколений в конечном счете будут определяться подготовленностью к восприятию этих средств со стороны пользователей, не занятых непосредственно разработкой или обслуживанием вычислительных машин и их программного обеспечения.

В этой работе предпринята попытка в систематизированной форме изложить широкому кругу специалистов, которые по роду своей деятельности используют или будут использовать современные вычислительные машины, возможности технических средств и программного обеспечения ЕС ЭВМ путем рассмотрения вопросов разработки, внедрения и эксплуатации прикладного программного обеспечения. Мы считаем этот вопрос важнейшим в подготовке пользователей, поскольку в нем концентрируется большая часть проблем внедрения и эффективной эксплуатации вычислительной техники в народном хозяйстве. Кроме того, подобный подход позволяет объективно определить объем и характер знаний о технических средствах и операционных системах, которые необходимы массовому пользователю.

Следует отметить, что чтение данной книги возможно без предварительного ознакомления с другими работами, в которых подробно рассмотрены вопросы технических средств и операционных систем ЕС ЭВМ. Однако для более глубокого понимания некоторых вопросов создания и применения прикладных программ читатель может обращаться к литературе, предложенной в библиографии с учетом ссылок, систематически используемых в тексте.

При определении круга вопросов, интересующих пользователей, мы исходили из опыта проведения семинаров и совещаний по эксплуатации ЕС ЭВМ и работы со специалистами, занятыми созданием автоматизированных систем управления на базе ЕС ЭВМ на различных предприятиях и в организациях. Именно в этой связи и с учетом данного опыта в книге излагаются некоторые вопросы разработки, внедрения и эксплуатации прикладных программ.

Таким образом, данную книгу нужно рассматривать как введение для пользователей.

В процессе создания и применения прикладного программного обеспечения возникают две важные задачи: разработка типовых прикладных программ и создание на базе этих программ эффективных систем обработки данных.

В настоящее время при создании системы обработки данных разработка прикладных программ является «узким местом». Затраты времени и средств, требуемые для такой разработки, иногда значительно превосходят те, которые необходимы для приобретения и установки комплекса технических средств. В связи с этим задача типизации прикладных программ, т. е. создание предпосылок для их массового распространения, приобретает большое значение. При этом подобная задача может решаться лишь с учетом специфики сфер применения и классов решаемых задач, а также с учетом возможностей и средств, представляемых той или иной вычислительной машиной или семейством ЭВМ.

С другой стороны, при использовании типовых прикладных программ в системах обработки данных возникает ряд задач, связанных с эффективной организацией этих систем. Современные вычислительные средства создают предпосылки для решения этих задач путем создания специальных программных систем. В настоящее время такие системы распространяются в различных сферах применения ЭВМ.

В условиях увеличения затрат на создание сложных программных систем и повышения трудоемкости этих работ разработку прикладной программы следует рассматривать как комплекс научно-исследовательских и опытно-конструкторских работ. Такой подход к созданию программы позволяет выявить и рассмотреть ряд организационно-экономических и технических задач проведения разработки.

Более детально принципы построения и использования программных систем в данной работе рассматриваются на примере пакета прикладных программ, предназначенного для решения задач сетевого планирования и управления. Выбор примера объясняется рядом причин. Во-первых, задачи оперативного управления исследованиями и разработками представляются одними из наиболее сложных задач управления. Рассматривая программную реализацию сетевых методов решения этих задач, мы имеем возможность обобщения некоторых принципов и методов применения ЭВМ в управлении. Во-вторых, сетевые методы управления представляют достаточно простой и наглядный аппарат, что позволяет сосредоточить внимание читателей на содержательной части вопросов, не отвлекаясь от основной темы изложения. В-третьих, программная реализация системы СПУ, описанная в этой книге, является типичной для задач данного класса.

Одним из наиболее существенных преимуществ ЕС ЭВМ для пользователей является возможность реализации таких режимов эксплуатации, как режимы разделения времени. Работа вычисли-

тельных машин в этих режимах является перспективным направлением использования ЭВМ. В этой связи рассмотрены некоторые вопросы построения прикладных программ для эксплуатации ЭВМ в указанных режимах. Более подробно описаны технические средства и программное обеспечение, реализующие соответствующие режимы работы вычислительных машин, поскольку данный вопрос, как нам кажется, еще недостаточно освещен в отечественной литературе.

Одна из основных идей, которую нам хотелось раскрыть в книге, заключается в том, что весьма сложная техническая система (ЕС ЭВМ) существенно упрощает процесс эксплуатации ЭВМ вообще, т. е. мы хотели показать, что менее сложные вычислительные системы второго поколения приводили к более сложным проблемам с позиции их использования, например, в системах управления. Сложность ЕС ЭВМ, как развитой системы программно-аппаратных средств, объясняется тенденцией к упрощению процессов массовой эксплуатации вычислительных машин. Однако эти тенденции могут найти практическое выражение лишь в ходе разработки и эксплуатации прикладных программ, использующих все те возможности, которые предоставляются ЭВМ третьего поколения. При этом отметим, что речь идет не об упрощении работы специалистов по применению вычислительной техники, а о возможности создания систем обработки данных, которые будут более доступны широкому кругу пользователей. Такой подход позволит повысить эффективность использования вычислительных машин и существенно расширить область их применения. В настоящее время изучение этой проблемы является одной из важнейших задач в области применения вычислительной техники. В этом плане данная работа может оказаться полезной и специалистам по применению ЭВМ, поскольку мы пытались сформулировать в ней некоторые важные требования к системам обработки данных и пути их реализации на базе ЕС ЭВМ.

Следует отметить, что, как и во всякой быстро развивающейся области, в сфере применения ЭВМ терминология еще окончательно не установилась. В связи с этим мы сочли возможным в некоторых случаях использовать термины, которые не имеют широкого распространения, однако отражают существо излагаемых вопросов. В частности, это относится к терминам «архитектура программы», «программная система», «режим работы», «режим эксплуатации» и т. п. Эти термины иногда встречаются в литературе, но обычно они носят вспомогательный характер и определяемые ими понятия подробно не обсуждаются. В связи с этим в некоторых случаях при введении терминов не делается специальных ссылок на литературу.

Мы надеемся, что эта книга в какой-то мере поможет широким кругам пользователей познакомиться с теми проблемами создания, освоения и использования прикладного программного обеспечения, без понимания которых невозможна эффективная эксплуатация ЕС ЭВМ.

Глава 1

СФЕРЫ ПРИМЕНЕНИЯ И ХАРАКТЕРИСТИКИ ПРИКЛАДНЫХ ПРОГРАММ

1. 1. СОСТАВ И ФУНКЦИИ СИСТЕМЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЭВМ

1. 1. 1. Значение программного обеспечения ЭВМ

Всю совокупность программ, разработанных и используемых для вычислительной машины или семейства ЭВМ, мы называем системой программного (математического) обеспечения.

Возможности вычислительной машины, обусловленные ее техническими характеристиками, в существенной мере определяют состав системы программного обеспечения этой машины с позиций количества используемых программ и сложности решаемых ими задач. В связи с этим ЭВМ, более совершенные в техническом отношении, имеют и более развитую систему программного обеспечения. С другой стороны, состав системы программного обеспечения любой распространенной вычислительной машины постоянно расширяется в связи с появлением все новых прикладных задач и сфер применения. Развитие вычислительной техники и расширение ее использования приводят к тому, что экономические пропорции в соотношении затрат на создание технических средств и программного обеспечения непрерывно изменяются в сторону повышения затрат на программное обеспечение (рис. 1)*.

Таким образом, важность изучения вопросов построения и эксплуатации программного обеспечения обосновывается как постоянным расширением области применения вычислительных машин, так и возрастающей ролью этих вопросов в развитии вычислительной техники.

* Рисунок взят из статьи: Boehm B. W. Software and Its Impact: A Quantitative Assessment. — "Datamation", 1973, vol. 19, № 5, p. 48—59. В дальнейшем цитируется как "Datamation", vol. 19, № 5, p. 48—59.

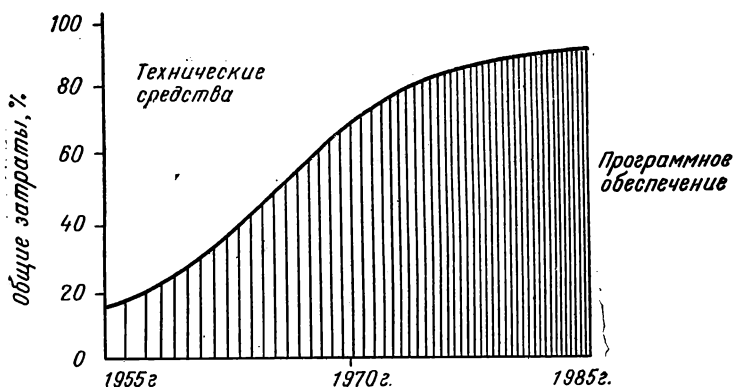


Рис. 1. Тенденции соотношения затрат на технические средства и программное обеспечение

1. 1. 2. Состав системы программного обеспечения

Система программного обеспечения любой ЭВМ неоднородна. При эксплуатации машины выделяются классы программ, отличающиеся друг от друга по распространенности в системах обработки данных у пользователей, эксплуатирующих ЭВМ данного типа, и по функциям, которые эти программы реализуют. С позиций этих двух факторов можно выделить четыре класса программ, в том или ином виде встречающихся в системе программного обеспечения

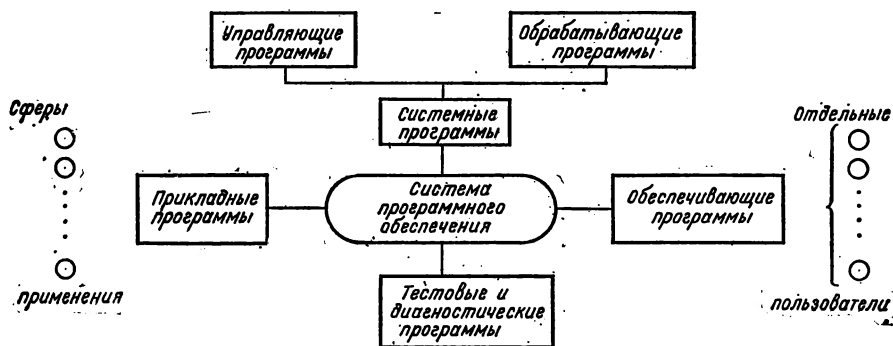


Рис. 2. Структура системы программного обеспечения

ния любых вычислительных машин: системные, прикладные, обеспечивающие или обслуживающие, тестовые и диагностические (рис. 2).

Ниже рассмотрен каждый класс программ и установлена некоторая взаимосвязь между ними.

Системные программы. Совершенствование средств и возможностей вычислительных машин выражается в постоянном развитии программного обеспечения. При этом в вычислительной технике программирование уже давно используется не только для эксплуатации ЭВМ пользователем, но и для обеспечения самого процесса функционирования ЭВМ. В связи с этим выделяется класс программ, предназначенных не для обработки данных пользователя, а например, для координации работы различных устройств ЭВМ, организации диалога оператора с вычислительной машиной, обеспечения помощи программисту в разработке и отладке новых программ и выполнения многих других функций. В этом смысле данный класс программ представляет непосредственное продолжение аппаратных средств ЭВМ. Эти программы являются неотъемлемой частью современной вычислительной машины и прямо или косвенно участвуют в обеспечении практически всех функций, выполняемых ЭВМ.

Комплекс таких программ, обеспечивающих функционирование и эффективную эксплуатацию ЭВМ, называется системным программным обеспечением.

Развитие системного программного обеспечения привело для ЭВМ третьего поколения к появлению крупных комплексов системных программ — операционным системам. Существует несколько целей создания операционных систем, как и системных программ вообще: увеличение производительности вычислительных систем, повышение производительности труда программиста и др. В основном первая цель реализуется управляющими, а вторая — обрабатывающими программами операционной системы (рис. 3) [17], [19].

Для эксплуатации современных вычислительных машин необходимо наличие операционной системы. Так, любая ЭВМ Единой системы, подготовленная к работе, должна быть оснащена одной из операционных систем, входящих в состав программного обеспечения ЕС ЭВМ [4], [17], [19]. На любой модели ЕС ЭВМ программа выполняется под управлением и при участии операционной системы. В этом смысле системное программное обеспечение в ЕС ЭВМ является наиболее распространенной составной частью общей системы программного обеспечения, так как оно должно входить в систему обработки данных любого пользователя, эксплуатирующего вычислительные машины этого семейства*.

Прикладные программы. Практически системные программы создают предпосылки для эффективного использования технических средств вычислительных систем, однако они не предназначены для непосредственной обработки данных по задачам пользователей. Эти функции реализуются с помощью прикладных программ,

* Операционные системы ЕС ЭВМ обеспечивают возможность выбора и настройки системных программ с учетом требований пользователя путем генерации [17], [19].

т. е. программ, ориентированных на решение конкретных задач в различных сферах применения (например, задач материально-технического снабжения на промышленном предприятии или задач численного решения нелинейных дифференциальных уравнений).

Любая прикладная программа, поскольку она решает определенную прикладную задачу, распространена в той сфере приме-

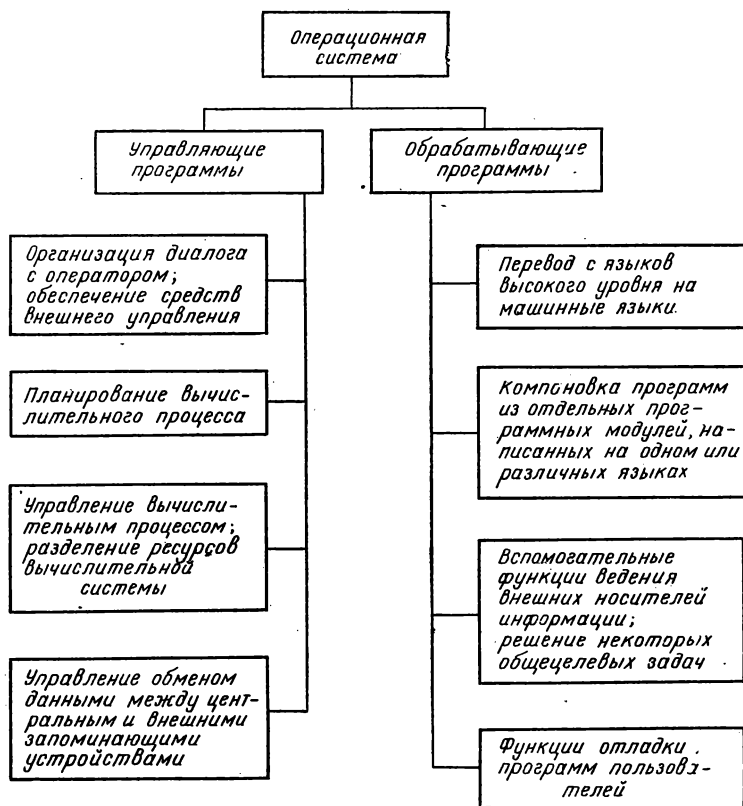


Рис. 3. Основные функции современной операционной системы

ния, где эта задача часто встречается у пользователей. В этом смысле любая отдельно взятая прикладная программа используется не так широко, как системная программа. Однако система обработки данных пользователя помимо системных всегда содержит и необходимые прикладные программы.

Обеспечивающие программы. В рамках каждой конкретной сферы применения прикладные программы решают лишь наиболее распространенные в этой сфере задачи. Однако, как правило, тот или иной набор прикладных программ не решает всех задач, необходимых отдельному пользователю. При отсутствии таких про-

грамм пользователь вынужден прибегать к самостоятельному программированию для решения возникших задач. Программы, разработанные пользователем для решения своих прикладных задач, мы называем обеспечивающими*.

Обеспечивающие программы, в отличие от прикладных, обычно не получают распространения за пределами конкретной системы обработки данных и того пользователя, для которых они созданы. С другой стороны, обеспечивающие программы, так же как и прикладные, предназначены для решения прикладных задач пользователей.

Обеспечивающие программы представляют по объему значительную долю всего программного обеспечения любой широко распространенной ЭВМ, однако зачастую они не рассматриваются как часть системы программного обеспечения по ряду причин, и в частности, из-за больших трудностей их изучения и классификации. С другой стороны, следует иметь в виду, что многие учебники по программированию, имеющиеся в нашей стране и за рубежом, посвящены именно вопросам создания обеспечивающих программ и не рассматривают специально вопросы создания прикладных и системных программ.

Тестовые и диагностические программы. Программы могут использоваться не только для перечисленных выше функций, но и для проверки правильности функционирования технических средств в различных режимах работы ЭВМ. С этой целью для современных вычислительных машин разрабатываются комплексы тестовых и диагностических программ, служащих для проверки правильности работы, поиска неисправностей в отдельных узлах и блоках вычислительных машин. Рассмотрение этих программ выходит за рамки данной книги.

1. 1. 3. Развитие системы программного обеспечения

В каждой сфере применения постоянно возникают новые задачи, которые в первое время проявляются как задачи отдельных пользователей и реализуются обеспечивающими программами. В дальнейшем, если задача получает широкое распространение в данной сфере применения, ее программные реализации совершенствуются в направлении создания универсальной программы, учитывающей специфику задач отдельных пользователей. При этом создание прикладной программы основывается на опыте разработки и эксплуатации многих обеспечивающих программ, предназначенных для решения данной задачи у различных пользователей.

* От американского термина «maintenance programming». Этот термин несколько шире и включает не только создание программ пользователем, но и модификацию имеющихся прикладных программ с учетом специфики задач пользователей. Однако такие модификации пользователь зачастую не может провести самостоятельно из-за сложности прикладных программ. В связи с этим задача создания обеспечивающих программ отделяется от задачи модификации. Реализация последней относится к деятельности по сопровождению программ, которая подробнее рассматривается в третьей главе.

Наличие прикладной программы позволяет расширить круг пользователей, реализующих машинное решение данной задачи. Однако не следует забывать, что разработка прикладной программы во многом зависит от той базы, которая была заложена соответствующими обеспечивающими программами. Таким образом, обеспечивающие программы составляют основу развития прикладного программного обеспечения, что является одной из главных причин их включения в систему программного обеспечения (рис. 3).

Естественно, что в различных сферах применения ЭВМ общие прикладные задачи встречаются гораздо реже, чем это имеет место для различных пользователей в рамках одной сферы применения. Однако в процессе развития различных сфер применения выявляются общие требования к вычислительным системам, которые реализуются отдельными компонентами прикладных программ или целыми программами. По мере накопления опыта разработки и эксплуатации прикладных программ эти требования находят выражение в развитии системного программного обеспечения. Прикладные программы, реализующие эти требования в различных сферах применения, служат основой для создания универсальных программ, т. е. для обеспечения того уровня общности, который необходим с учетом массового распространения системного программного обеспечения. В этом смысле развитие сфер применения и прикладных программ представляет базу для разработки новых системных программ и системного программного обеспечения в целом. В качестве примера можно привести возникновение систем управления данными, как части операционных систем. Системы управления данными [17], [19] предназначены для эффективного обмена данными между основной и внешней памятью ЭВМ, а также для эффективной организации данных на внешних носителях информации. Разработка этой системы тесно связана с проблемой организации сбора, хранения и поиска данных, которая имеется в различных сферах применения (в первую очередь экономических), где большие объемы обрабатываемой информации представляют основную сложность решения задачи.

Таким образом, выделение отдельных составляющих системы программного обеспечения позволяет рассматривать развитие этой системы как процесс, при котором программная реализация решения задачи постепенно достигает того уровня общности (универсальности), который определяется степенью распространения данной задачи у пользователей. Непрерывность развития программного обеспечения обуславливается постоянным появлением новых прикладных задач у пользователей.

В работе рассматриваются прикладные программы, принципы их построения и эксплуатации, а также их место в системе программного обеспечения вычислительных машин третьего поколения.

Практической основой для некоторых положений, сформулированных в этой работе, послужил анализ отечественных и зарубежных прикладных программ. Количественные результаты анализа приводятся в приложении 1.

1.2. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ. ОСНОВНЫЕ ПОНЯТИЯ И ХАРАКТЕРИСТИКИ

Основа прикладного программного обеспечения. Каждая сфера применения ЭВМ определяет специальные требования к созданию прикладных программ, а качество программных реализаций в свою очередь зависит от возможностей, предоставляемых вычислительной машиной в распоряжение программиста. В настоящее время достаточно сложно охватить весь опыт использования ЭВМ в различных сферах применения с учетом специфических требований, определяющих направление развития прикладных программ. Выделим лишь наиболее распространенные сферы применения ЭВМ и рассмотрим для них особенности разработки и эксплуатации прикладных программ. В процессе широкого внедрения ЕС ЭВМ разработка именно этих программ представляет наибольший интерес, поскольку они определяют эффективность массового использования вычислительных машин. Однако подходы к анализу прикладного программного обеспечения, принятые здесь, могут быть использованы и для других сфер применения ЭВМ.

В любой сфере применения вычислительных машин встречаются программы как однократного, так и регулярного использования*. Соотношение тех и других программ по объемам затрат на разработку и количеству используемого машинного времени различно для разных сфер применения. Поскольку прикладные программы регулярного использования представляют базовую часть прикладного программного обеспечения ЭВМ и в них концентрируются общие требования и тенденции развития прикладного программного обеспечения, эти программы являются основным предметом рассмотрения данной книги.

Интегрированные системы обработки данных. Наряду с расширением области применения вычислительных машин наблюдается процесс усложнения самих решаемых задач. В настоящее время происходит массовый переход от решения задач с большим объемом вычислений, но простыми алгоритмами, к задачам, в которых значительное число вычислительных процедур сочетается со сложностью и многовариантностью самих методов и алгоритмов решения. Так, при автоматизации управления народнохозяйственными объектами осуществляется переход от решения отдельных задач к созданию комплексных автоматизированных систем управления, охватывающих существенную часть функций и задач руководства.

В связи с этими изменениями в характере использования вычислительных машин в настоящее время качественно изменяются

* В любой системе обработки данных есть задачи, решаемые регулярно, и задачи, систематическое решение которых не требуется. Программные реализации задач первой группы мы относим к регулярно используемым программам, второй — к программам однократного применения.

Отметим, что программы однократного использования обычно относятся к обеспечивающим программам. Основу регулярно используемых программ образуют прикладные программы. В то же время многие обеспечивающие программы могут регулярно применяться в той или иной системе обработки данных,

формы и методы обработки данных. Если раньше процесс эксплуатации ЭВМ сводился к разработке и применению набора программ для решения отдельных задач пользователей, то сейчас все в большей степени проявляется тенденция к созданию крупных интегрированных систем обработки данных на базе программно-аппаратных средств современных ЭВМ.

Под системой обработки данных понимается комплекс технических средств ЭВМ, системного, прикладного и обеспечивающего программного обеспечения, реализующий решение прикладных задач. В тех случаях, когда это не может вызвать разночтений, мы будем для краткости употреблять термин «система обработки данных» в более узком смысле, обозначая им прикладное программное обеспечение системы обработки данных.

Основная особенность интегрированных систем обработки данных состоит в том, что они представляют собой комплексы взаимосвязанных прикладных программ, ориентированных на решение определенной сложной проблемы. При этом взаимосвязь программ в рамках интегрированной системы отражает сложность и большой объем самой решаемой проблемы, что не позволяет реализовать ее решение с помощью одной программы. Так, в процессе создания АСУ промышленным предприятием требуется решение задач оперативного управления, материально-технического снабжения, бухгалтерского учета и многих других. Поскольку решение этих задач в условиях реального предприятия невозможно без учета их взаимного влияния, эти взаимосвязи должны найти свое отражение в прикладных программах, реализующих решение перечисленных задач в рамках интегрированной системы обработки данных.

Пакеты прикладных программ. В настоящее время интегрированные системы обработки данных определяют современный уровень эксплуатации вычислительных машин. В связи с этим основным инструментом обработки данных становится не отдельная программа, а элемент интегрированной системы — комплекс программ, предназначенный для решения определенной задачи или класса задач. Такой комплекс программ вместе с документацией, необходимой для его установки и эксплуатации, называется пакетом прикладных программ. В дальнейшем наряду с термином «пакет прикладных программ» мы для краткости будем употреблять термины «прикладная программа» и «пакет».

Пакеты прикладных программ должны удовлетворять требованиям, предъявляемым к программам пользователя теми операционными системами, под управлением которых они функционируют, а программы, входящие в состав пакета, должны разрабатываться с учетом стандартных соглашений, принятых для соответствующих операционных систем [17]. В рамках ЕС ЭВМ пакеты должны разрабатываться с учетом возможностей их функционирования как на различных конфигурациях моделей Единой системы, так и на различных конфигурациях операционной системы.

С позиций использования вычислительных машин пакеты прикладных программ представляют собой базу для построения интегрированных систем обработки данных. В плане объема и сложности задач, решаемых пакетами, примерами могут служить пакеты прикладных программ для реализации задач линейного программирования, материально-технического снабжения, для реализации

статистических методов обработки данных и т. д.

Основная особенность пакетов прикладных программ в отличие от программ, которые получили широкое распространение для ЭВМ второго поколения, состоит в том, что пакет ориентируется на решение задач большего объема и зачастую большей сложности. С позиций эффективного построения прикладного программного обеспечения для интегрированных систем и эффективного использования вычислительных средств ЕС ЭВМ целесообразно даже простые задачи пользователей рассматривать как часть некоторой более общей задачи и реализовывать решение этих частных задач в рамках более крупных программных комплексов — пакетов прикладных программ.

Характерными примерами таких комплексов программ служат библиотеки программ, широко распространенные для

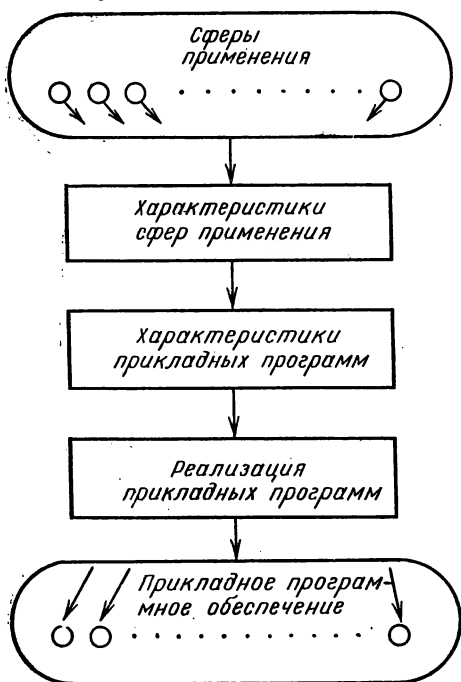


Рис. 4. Логика реализации прикладного программного обеспечения

ЭВМ второго поколения. Однако следует различать понятия пакета прикладных программ и библиотеки программ. Понятие «пакет прикладных программ» служит для определения обособленных элементов, из которых состоит система прикладного программного обеспечения. При этом внутренняя реализация различных пакетов может быть неодинаковой. Понятие «библиотека программ» описывает один из возможных принципов внутренней реализации пакета и его эксплуатации.

Различные сферы применения вычислительных машин предъявляют особые требования к структуре и принципам использования пакетов прикладных программ. Следовательно, общая логика изучения пакетов может сводиться к анализу условий их применения и описанию на базе такого анализа возможных вариантов внутренней реализации тех или иных программ. С этой целью мы выделим

для анализа некоторые важные характеристики сфер применения ЭВМ и характеристики пакетов прикладных программ (рис. 4). Далее рассмотрим некоторые связи между этими характеристиками.

Под характеристиками сфер применения ЭВМ подразумеваются такие особенности классов прикладных задач, которые определяют требования к программной реализации этих задач. Характеристики пакетов прикладных программ включают особенности программ, которые обеспечивают удовлетворение этих требований. Такие особенности касаются внутренней реализации и способов использования программ. Таким образом, уже в определении характеристик сфер применения ЭВМ и прикладных программ отражается связь между требованиями к пакету и его реализацией.

1. 2. 1. Характеристики сфер применения ЭВМ

Ниже рассмотрены две важные характеристики сфер применения ЭВМ: специфика сферы применения, т. е. характер решаемых в ней прикладных задач, и квалификация пользователей в данной

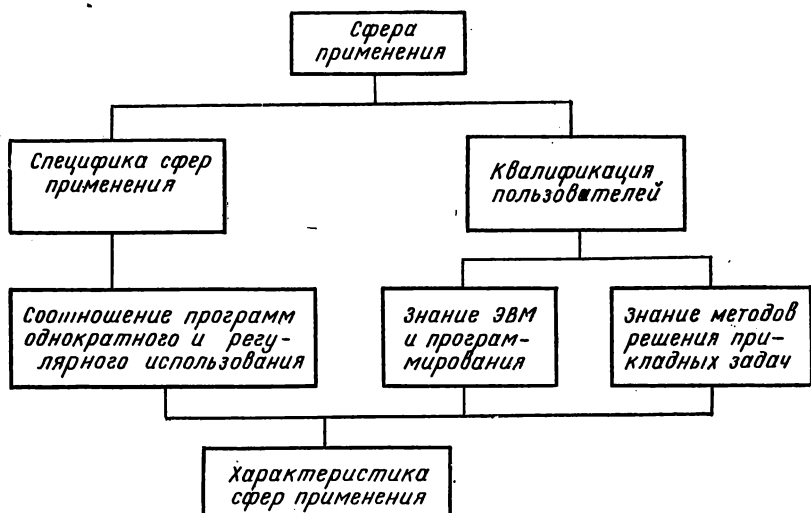


Рис. 5. Анализ характеристик сфер применения ЭВМ

сфере применения с позиций знания элементов программирования и алгоритмов решения задач (рис. 5).

Отметим, что пользователями мы называем специалистов в той или иной сфере применения, целью которых является получение результатов решения прикладных задач с помощью ЭВМ.

Специфика сфер применения. Программы однократного и регулярного применения. Одним из главных факторов, определяющих специфику отдельных сфер применения ЭВМ, является соотношение объемов программной продукции однократного применения и

регулярного использования* для прикладных задач каждой сферы применения. Эти соотношения можно было бы измерять как соотношение общих объемов затрат на разработку тех и других программ или как соотношение общего числа машинных инструкций. Точное количественное выражение подобной характеристики для сферы применения вряд ли возможно, однако даже расчет таких соотношений на базе какой-либо крупной системы обработки данных позволил бы в определенной мере характеризовать тенденции использования ЭВМ в той сфере применения, решению задач которой служит эта система. Для сравнения отдельных крупных сфер применения в данном случае вполне достаточно качественного анализа. В частности, при использовании ЭВМ в научных и инженерных расчетах соотношение объемов программной продукции однократного и регулярного применения должно быть гораздо выше, чем при использовании вычислительных машин в автоматизированных системах управления (как организационными, так и технологическими процессами).

Действительно, применения ЭВМ для научных разработок характеризуются постоянным появлением новых задач, требующих специальных программ для своего решения. При этом такие программы могут и не быть использованы в дальнейшем ввиду индивидуальности самой прикладной задачи. В этом смысле задачи АСУ отличаются большей «стабильностью», и однажды составленная программа зачастую отражает потребность в регулярном решении определенной задачи управления.

Естественно, что при дальнейшей конкретизации прикладных задач в рамках научных и инженерных применений, с одной стороны, и применений ЭВМ в системах управления, с другой стороны, специфика отдельных сфер применения определится более точно и приведенное общее утверждение будет преломляться в них различным образом.

1. 2. 2. Квалификация пользователей ЭВМ

Вторым важным вопросом при оценке направлений развития прикладного программного обеспечения является квалификация пользователей. Она определяется, во-первых, степенью знания вычислительной техники вообще и вопросов программирования на ЭВМ в частности и, во-вторых, уровнем знания алгоритмов решаемых задач. С позиций обоих факторов можно выделить три категории пользователей: программисты-пользователи, квалифицированные пользователи и широкий пользователь **.

* Отметим, что к регулярно используемым относятся те программы, которые требуются не только для некоторой конкретной системы обработки данных, но достаточно широко распространены в системах обработки данных пользователей определенной сферы применения.

** Следует иметь в виду, что приведенные термины характеризуют определенные категории пользователей только с позиций применения ими вычислительной машины для решения прикладных задач.

Программисты-пользователи. Как и при любой специализации, уровень профессиональных знаний (уровень знания вычислительной техники и программирования) различен для разных программистов-пользователей. Однако в целом определяющим для пользователей этой категории является то, что они специализируются на создании программ для решения прикладных задач. В связи с этим программист-пользователь отличается от пользователей других категорий более глубоким знанием вопросов программирования на ЭВМ и алгоритмов решаемых прикладных задач. Практические знания в области программирования у этого круга пользователей проявляются как знание одного или нескольких языков программирования (как машинно-ориентированных, так и высокого уровня), общих принципов программирования на ЭВМ определенного типа, а также (и это главное) как знание и опыт использования общих принципов и методов разработки, отладки прикладных программ. Последнее позволяет программистам-пользователям разрабатывать эффективные прикладные программы регулярного использования, требования к которым значительно выше, чем к программам однократного применения. Говоря о детальном знании программистом алгоритмов решаемой задачи, мы имеем в виду знание именно формальных алгоритмов, а не методов решения прикладных задач в данной сфере применения. Программист выступает как специалист по переводу алгоритма решения задач на язык вычислительных машин с целью создания эффективной прикладной программы, а не как специалист в данной области знаний. В связи с этим он в разное время может быть занят разработкой программ для различных сфер применения ЭВМ, хотя в этом случае, конечно, утрачивается некоторый опыт, обусловленный спецификой сферы применения*.

Квалифицированные пользователи. С позиций знания вопросов программирования категория данных пользователей характеризуется знанием общих принципов программирования и каких-либо языков программирования. Это позволяет пользователю данной категории разрабатывать программы однократного применения для непосредственного использования вычислительной машины. Знание прикладных задач данной сферы применения проявляется у пользователей этой категории как свободное владение общими принципами и методами решения задач определенного класса.

В основном категория квалифицированных пользователей включает специалистов в научных и инженерных областях знаний. Здесь сведения по вычислительной технике и общим принципам ее использования в настоящее время уже достаточно широко распространены. Это обусловлено в первую очередь непрерывным появлением новых задач и преобладанием в данных сферах применения программ однократного использования, что существенно увеличи-

* В зарубежной литературе по применению вычислительной техники приняты термины «научный программист», «коммерческий программист» и т. д. При этом обычно они характеризуют не знания специалиста в данной сфере применения, а опыт реализации прикладных программ для данной сферы.

вает потребность в программировании и как следствие — потребность в кадрах программистов. Практически эта потребность настолько велика, что не может быть удовлетворена только за счет квалифицированных программистов-пользователей, а требует привлечения к программированию специалистов, занятых в соответствующих сферах применения. С другой стороны, высокий уровень знания прикладных задач и хорошее владение аппаратом, выработанным для их решения, создает этому кругу специалистов необходимые предпосылки для работы с ЭВМ.

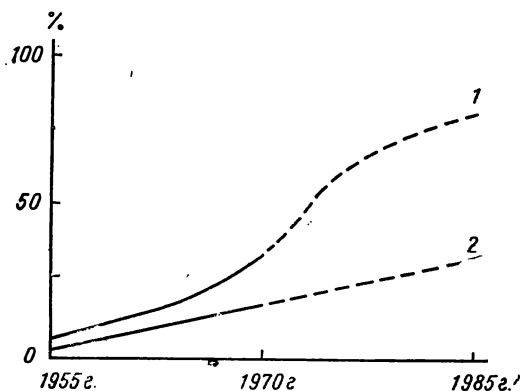


Рис. 6. Тенденции роста пользователей ЭВМ в соотношении с ростом квалифицированных пользователей и специалистов:
1 — общее число пользователей; 2 — число квалифицированных пользователей и специалистов

танным для их решения, создает этому кругу специалистов необходимые предпосылки для работы с ЭВМ.

Широкий пользователь. Уровень знаний широкого пользователя в области вычислительной техники и использования ЭВМ ограничивается лишь знанием самых общих принципов. Знание прикладных задач может проявляться здесь как понимание их содержательной части или только как умение использовать результаты их решения в практической деятельности.

Пользователями этой категории являются, например, руководители предприятий и планово-экономических служб в условиях внедрения автоматизированных систем управления. Именно в связи с внедрением АСУ, а также дальнейшим расширением области применения ЭВМ широкий пользователь должен в ближайшее время составить наиболее массовую группу пользователей вычислительных машин (рис. 6) *. Безусловно, при функционировании АСУ на предприятиях к специалистам, занятым в сфере управления, предъявляются определенные требования, однако по объему знаний в области вычислительной техники и программирования эти требования значительно ниже, чем для двух других категорий пользователей.

Обращаясь вновь к логике реализации прикладного программного обеспечения (рис. 4), рассмотрим теперь характеристики прикладных программ и связь требований сфер применения с возможными вариантами реализации прикладных программ.

1. 2. 3. Характеристики пакетов прикладных программ

Особенности внутренней реализации программы и ее использования в существенной мере определяются требованиями соответствующих сфер применения и, в частности, рассмотренными выше ха-

* Рисунок взят из "Datamation", vol. 19, № 5 p. 48—59.

рактическими сферами. Ниже рассмотрены две характеристики прикладных программ, отражающие эти связи.

Решение прикладной задачи связано с применением тех или иных методов решения. При этом обычно определенный метод может быть применен для решения целого класса прикладных задач. И наоборот, одна и та же прикладная задача может решаться с применением различных методов.

С позиций разработки прикладных программ целесообразно различать сам математический метод и его применение к решению той или иной прикладной задачи. В применении математического метода к решению конкретной прикладной задачи возможности самого метода обычно сужаются, ориентируясь на данную задачу. Иначе говоря, математический метод многообразнее любого своего применения.

Прикладная программа как инструмент для решения прикладных задач с использованием ЭВМ может быть ориентирована на реализацию определенного метода решения задач или на его применение к реализации отдельной важной прикладной задачи. Ориентация прикладной программы — первая из рассматриваемых ниже характеристик программ. Вторая характеристика — степень автоматизации процесса вычислений, обеспечиваемая прикладной программой, — в существенной мере отражает простоту ее использования (рис. 7).

Методо- и проблемно-ориентированные пакеты прикладных программ. Специфика сфер применения ЭВМ, выражающаяся в соотношении объемов однократно и регулярно используемой программной продукции, определяет направление разработки прикладных программ для отдельных сфер применения.

В тех случаях, когда в данной области непрерывно появляются новые прикладные задачи, решение которых требует применения одних и тех же математических методов, удобно создавать пакеты прикладных программ, реализующие эти методы, а не их применения к решению отдельных задач (рис. 8, а). Такие пакеты называются методо-ориентированными (например, пакеты программ для решения задач численного анализа, статистических задач и т. д.).

Большое разнообразие прикладных задач, которые могут быть решены с использованием того или иного метода, и постоянное появление новых задач, решаемых этим методом, не позволяют реализовать все возможные алгоритмы решения различных задач в рам-

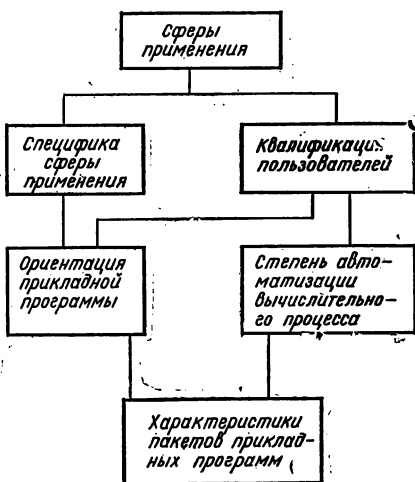


Рис. 7. Анализ характеристик пакетов прикладных программ

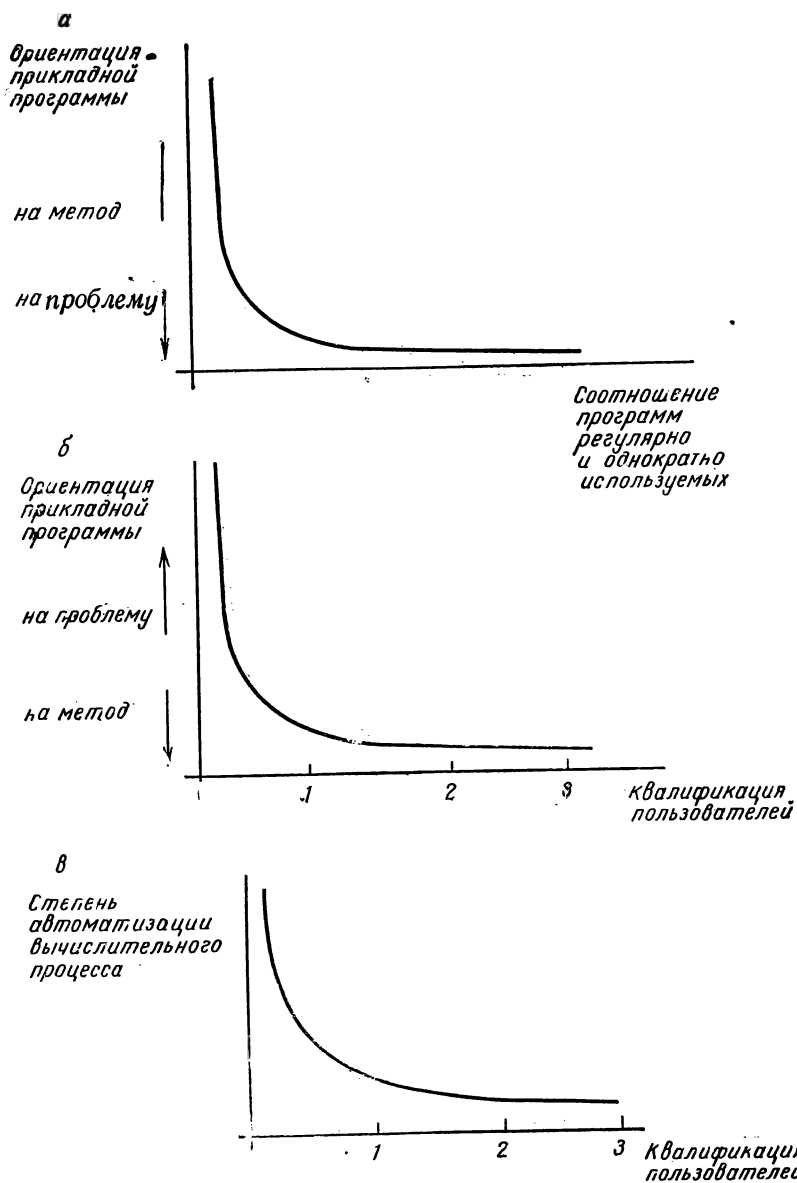


Рис. 8. Схемы соотношения характеристик сфер применения и пакетов прикладных программ*.

1 — широкий пользователь; 2 — квалифицированный пользователь; 3 — программист-пользователь

* Этот и следующий рисунки носят иллюстративный характер.

как одного пакета. В связи с этим методо-ориентированные пакеты прикладных программ в большинстве случаев предназначаются для упрощения работы пользователя, а не для непосредственного решения прикладной задачи. Отметим, что каждый конкретный методо-ориентированный пакет используется для упрощения программной реализации именно тех прикладных задач, которые решаются с применением соответствующего математического метода (например, метода статистического моделирования) и для других задач его применение нецелесообразно.

В случае, когда большому числу пользователей требуется регулярное решение определенной прикладной задачи, целесообразна программная реализация ее решения в виде отдельного пакета прикладных программ. Пакеты прикладных программ, предназначенные для реализации подобных задач, называются проблемно-ориентированными.

В отличие от методо-ориентированных проблемно-ориентированные пакеты прикладных программ непосредственно реализуют алгоритмы решения задач пользователей. При этом в рамках одного пакета можно обеспечить программную реализацию большого числа алгоритмов для учета особенностей решения данной задачи различными пользователями.

В качестве примера проблемно-ориентированных пакетов прикладных программ можно привести пакет для решения задач планирования загрузки производственных мощностей мелкосерийных предприятий и опытного производства в крупных научно-исследовательских институтах. Вообще в такой сфере применения ЭВМ, как система управления, где большинство задач (например, оперативное управление, материально-техническое снабжение, бухгалтерский учет и т. п.) решается непрерывно, проблемно-ориентированные программы получают широкое распространение.

Некоторые проблемно-ориентированные программы ведут обработку данных без применения каких-либо математических методов. В частности, решение задач бухгалтерского учета на предприятии в большинстве случаев не требует сложных расчетов, однако сопряжено с обработкой крупных массивов данных и представлением различных видов отчетов. Наличие развитых средств формирования всевозможных выходных документов является характерной особенностью проблемно-ориентированных программ.

В тех случаях, когда проблемно-ориентированная программа основана на применении определенных математических методов, целью пакета является не реализация этих методов, а их применение к решению определенной прикладной задачи. Основой для создания проблемно-ориентированных программ могут служить методо-ориентированные программы, обеспечивающие простую реализацию математических методов, необходимых для решения соответствующих задач. Например, при разработке пакета программ для планирования загрузки производственных мощностей можно использовать программу, реализующую методы статистического моделирования.

В каждом конкретном применении возможности математического метода не реализуются полностью. Особенности применения метода к решению одной прикладной задачи могут не проявляться при использовании его для решения другой задачи. В связи с этим использование проблемно-ориентированной программы, предназначенной для реализации первой задачи, при решении второй может быть неэффективным. Так, в частности, нецелесообразно использование прикладной программы планирования загрузки производственных мощностей в качестве инструмента при решении задачи оценки производительности действующих вычислительных систем, хотя эта задача может решаться с применением того же самого математического аппарата.

Приступая к разработке прикладной программы, необходимо заранее определить ее ориентацию на реализацию определенных математических методов или на их применение к решению конкретной прикладной задачи. Важным фактором при выборе направления реализации программы является квалификация пользователей, для которых она предназначена. Как правило, применение методо-ориентированных программ не избавляет пользователя от необходимости программирования для решения конкретной прикладной задачи, а лишь упрощает процесс программирования, обеспечивая удобные средства для описания алгоритма решения.

Проблемно-ориентированные программы в большинстве случаев исключают для пользователей необходимость программирования. Поэтому наличие в ряде сфер применения, и в частности в сфере управления, широкого пользователя как основного пользователя вычислительной машины является важным аргументом в пользу разработки в таких областях проблемно-ориентированных пакетов прикладных программ.

Таким образом, можно констатировать целесообразность применения методо-ориентированных пакетов прикладных программ для программистов-пользователей и квалифицированных пользователей и проблемно-ориентированных программ для широкого пользователя. Однако это не исключает возможности использования программ различной ориентации в одной и той же сфере применения. Существует целый ряд прикладных программ общего назначения, одинаково распространенных в различных сферах применения ЭВМ. Кроме того, наличие в таких сферах применения, как автоматизированные системы управления, большого числа программистов-пользователей предполагает применение методо-ориентированных прикладных программ в качестве базы для разработки программ АСУ. Связь характеристик сфер применения и характеристик программ проявляется в процессе эксплуатации ЭВМ лишь как общая тенденция (рис. 8, б).

Отметим также, что разделение прикладных программ на методо- и проблемно-ориентированные в определенной мере условно. Так, иногда одна и та же программа пользователям различных категорий может представляться по-разному ориентированной. Одна-

ко такое разделение программ считается полезным в методическом отношении и мы будем использовать его в дальнейшем.

Степень автоматизации вычислительного процесса. Применение прикладной программы связано с определенными требованиями к знаниям пользователей в области программирования. Эти требования различны для разных программ. В одних случаях использование прикладных программ невозможно без дополнительного программирования, в других — оно предполагает знание некоторых особенностей программы и, следовательно, принципов программирования вообще. Соответствующим образом разработанные программы могут обеспечивать автоматическую организацию процесса вычислений по достаточно простому запросу, составление которого не требует от пользователя каких-либо знаний в области программирования. Например, кассир, занятый продажей билетов на аэровокзале, может совсем ничего не знать о сложной программе, обеспечивающей автоматический поиск свободных мест на определенный рейс и выдачу требуемого билета. Он владеет лишь простыми правилами формулирования запроса, который дает программе исходную информацию для работы*.

Таким образом, степень автоматизации вычислительного процесса определяется степенью участия пользователя в организации процессов подготовки и выполнения программы. Чем выше степень участия пользователя, тем ниже степень автоматизации, и наоборот. Различные прикладные программы обеспечивают разную степень автоматизации вычислительного процесса. Естественно, чем выше степень автоматизации вычислительного процесса, обеспечиваемая прикладной программой, тем проще эксплуатация этой программы.

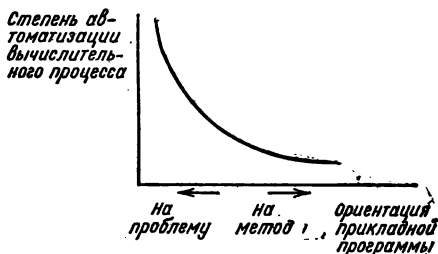


Рис. 9. Схема соотношения характеристик пакетов прикладных программ

Не следует думать, что для любой программы простота эксплуатации является важнейшим критерием оценки ее эффективности. В процессе разработки прикладной программы должна быть обеспечена такая степень автоматизации вычислительного процесса, которая соответствует требованиям использования этой программы. Оценивая необходимую степень автоматизации при разработке, нужно учитывать, с одной стороны, категорию пользователей, для которых предназначена программа, а с другой — тот факт, что разработка программ с более высокой степенью автоматизации вычислительного процесса требует обычно больших затрат времени

* Задача автоматизации вычислительного процесса обсуждается, например, в статье Б. А. Загацкого и Е. П. Михеевой «Некоторые проблемы разработки прикладных программ». — В сб.: Алгоритмы и организация решения экономических задач. Вып. 6. М., «Статистика», 1975.

и средств. Естественно, что основным потребителем программ, простых в эксплуатации, должен быть широкий пользователь, поскольку здесь высокая степень автоматизации является важнейшим условием использования ЭВМ (рис. 8, в).

Взаимосвязь характеристик пакетов прикладных программ. Отметим общие тенденции в ориентации пакетов прикладных программ и автоматизации вычислительного процесса для различных категорий пользователей, можно установить соответствие между этими двумя важными характеристиками. В целом для проблемно-ориентированных прикладных программ характерна более высокая степень автоматизации вычислительного процесса. Это положение представляет важную тенденцию для оценки развития прикладного программного обеспечения. Однако оно не отрицает возможности реализации в отдельных случаях методов ориентированных программ с высокой степенью автоматизации. Более детальный анализ потребует выяснения специфики конкретной исследуемой сферы применения и класса прикладных задач. На рис. 9 показаны общие тенденции взаимосвязи рассмотренных характеристик прикладных программ.

1. 3. ОСНОВНЫЕ ФОРМЫ ПРОГРАММИРОВАНИЯ ЗАДАЧ ОБРАБОТКИ ДАННЫХ

1. 3. 1. Архитектура прикладных программ

Программы, предназначенные для решения одной и той же задачи, могут существенно различаться по внутренней реализации. Иногда достаточно сложно определить, какая из разработанных программ более эффективна. Важнейшую роль в оценке эффективности программ играют требования той сферы применения и того пользователя, который будет в дальнейшем эксплуатировать эту программу. Необходимо строить программу таким образом, чтобы она была не только применима для решения данной задачи, но и наиболее полно отвечала условиям ее эксплуатации.

Важнейшим понятием, определяющим связь характеристик программы с ее реализацией, является структура программы, т. е. общие принципы построения программ пакета, его функционирования и эксплуатации. Наряду с термином «структура» далее как синонимы используются термины «архитектура» и «форма программирования».

Существующий опыт использования ЭВМ позволяет выделить три основные группы программ с точки зрения их структуры:

- библиотеки программ;

- системы программирования с использованием специализированных языков;

- программные системы.

Мы допускаем, что ряд программ, используемых в настоящее время, может выходить за пределы данной классификации или занимать промежуточное положение. Однако для дальнейшего анализа

прикладных программ и практики их реализации выделим и рассмотрим эти три класса программ, которые достаточно полно охватывают всю систему прикладного программного обеспечения.

Библиотека прикладных программ. Библиотека прикладных программ представляет собой традиционную форму программирования для ЭВМ первого и второго поколений, а также широко используется при работе с ЭВМ Единой системы. Для первых поколений вычислительных машин библиотеки программ составляли основу систем обработки данных.

С точки зрения структуры библиотека прикладных программ представляет собой набор отдельных программ, предназначенных для решения некоторого класса задач, или универсальный набор, применимый для решения различных, не связанных между собой задач. Каждая программа из библиотеки имеет самостоятельное значение и применима для решения некоторой, обычно достаточно простой задачи. В качестве примера можно привести библиотеки программ для вычисления приближенных значений элементарных функций (таких, как $\sin x$, $\operatorname{tg} x$, $\sqrt{x}$ и т. п.), которыми была оснащена практически любая ЭВМ первого поколения. Отметим еще мощную библиотеку научных программ (SSP), разработанную для вычислительных машин системы IBM/360 и представляющую набор программ, реализующих важнейшие математические методы [25].

В процессе эксплуатации библиотеки пользователь либо непосредственно применяет отдельные программы для решения небольших прикладных задач, либо использует программы библиотеки для реализации отдельных частей общей задачи, которую он решает. Таким образом, библиотека программ служит в основном вспомогательным средством для решения задач пользователей.

Подобная форма программирования эффективно используется для создания методо-ориентированных пакетов прикладных программ. При этом отдельные программы или небольшие группы программ предназначаются для реализации определенного метода. Например, библиотека программ, реализующих статистические методы обработки данных, может содержать программы корреляционного, регрессионного, дисперсионного анализа и др.

Хотя библиотека программ может разрабатываться достаточно длительное время в несколько этапов, такая разработка все же является целенаправленной в смысле логического единства программ библиотеки. Это единство выражается в общности используемого языка программирования, в общности самих решаемых задач и т. д. Следует различать пакет прикладных программ, реализованный в форме библиотеки, и архив программ, сформировавшийся в процессе длительной эксплуатации конкретной системы обработки данных. Архив, как правило, содержит программы, необходимые в работе данному пользователю в конкретной системе обработки данных, и в большинстве случаев не удовлетворяет потребности широкого круга пользователей.

Преимущество библиотеки программ как формы программирования состоит в возможности ее расширения путем добавления новых программ. Однако применение библиотек ограничено, поскольку при работе с ними необходимы достаточно глубокие знания основ программирования и принципов функционирования программного обеспечения, чего, безусловно, нельзя требовать от наиболее массовой группы пользователей — широкого пользователя. Применение библиотек пользователями этой категории возможно лишь для решения достаточно простых задач, реализуемых отдельно взятыми программами.

Таким образом, хотя библиотеки программ представляют хорошую базу для работы программистов-пользователей и квалифицированных пользователей, они не получают массового распространения в тех случаях, когда прикладные задачи должны решаться широким пользователем.

Системы программирования с использованием специализированных языков. Эта форма программирования применима для разработки прикладных программ, поскольку любой специализированный язык имеет определенную четко выраженную область применения, в которой его использование обеспечивает существенное упрощение процесса программирования.

Специализированные языки программирования, например СИМСКРИПТ и GPSS [38—40], предназначенные для решения многочисленных задач статистического моделирования, MATLAN—для работы с матрицами и другие, получили развитие для ЭВМ второго и третьего поколений. Они отличаются от проблемно-ориентированных языков (Алгола, Фортрана и др.) специфическими функциями отдельных операторов языка, определяемыми характером решаемых задач. Практически с помощью специализированного языка можно описывать алгоритм решения задачи на уровне укрупненной блок-схемы, а не детального алгоритма решения, как того требует применение языков высокого уровня. В остальном использование специализированных языков программирования идентично применению проблемно-ориентированных языков.

По своей реализации системы программирования с использованием специализированных языков могут отличаться друг от друга, и мы выделяем эту форму программирования, в первую очередь учитывая единство в принципах применения подобных программ.

Специализированные языки широко используются для решения различного рода задач в определенных сферах применения. Решение задачи с помощью такой программы обеспечивается некоторой совокупностью методов, последовательность и форма применения которых описываются алгоритмом решения на специализированном языке. Это свойство является существенным признаком методо-ориентированных прикладных программ. Система программирования с использованием специализированного языка требует от пользователей хорошего знания метода решения прикладных задач и используемого специализированного языка. В связи с этим данная

форма программирования ориентирована на программистов-пользователей и квалифицированных пользователей.

Программные системы. Прикладные программы в форме программных систем разрабатывались уже для ЭВМ второго поколения, однако широкое распространение эта форма программирования задач обработки данных получила лишь для ЭВМ третьего поколения. Программная система представляет собой комплекс

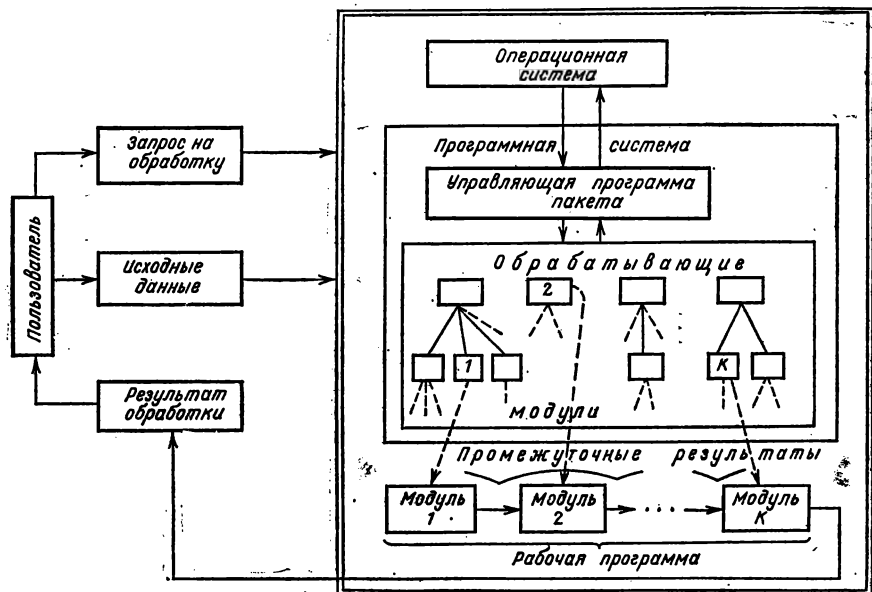


Рис. 10. Решение задачи с использованием программной системы

программ, называемых программными модулями*, и предназначена для решения определенной, сравнительно большой по объему задачи. Примерная структура пакета, имеющего архитектуру программной системы, и принцип его работы показаны на рис. 10.

Обращение к программной системе реализуется в форме запроса, который содержит требования к обработке данных и непосредственно данные. Обработкой данных по запросам руководит управляющая программа пакета (программной системы), которая для работы на ЭВМ третьего поколения обычно реализуется в форме описания решаемой задачи на языке управления заданиями [17], [19]. В этом случае управляющая программа пакета практически реализуется средствами операционной системы. Взаимосвязь отдельных модулей, реализованных в программной системе, обеспечивает воз-

* Понятие «программный модуль» отражает в данном случае иерархию в структуре программной системы и может включать, например, программную секцию как определенный уровень модульности при программировании на языке Ассемблера [23], [27].

возможность организации сложной обработки данных средствами языка управления заданиями. В случаях, когда управление решением задачи реализуется по более сложному алгоритму, управляющая программа системы разрабатывается как отдельная программа.

Непосредственную обработку данных ведут обрабатывающие модули пакета. Каждый программный модуль в свою очередь может разделяться на более мелкие модули (рис. 10). В подобной структуре должны быть четко определены функции конкретных модулей по обработке данных. Для достижения такой определенности и обособленности отдельных частей общего алгоритма решения этот алгоритм разбивается на достаточно мелкие этапы решения и каждый этап программно реализуется некоторым модулем. Группа этапов алгоритма, логически связанных между собой, образует программу, представляющую шаг решения задачи*.

При решении прикладной задачи пользователь располагает языковыми средствами программной системы, предназначенными для формулирования требований к обработке и описания вводимых данных. Обычно такие средства просты и вполне доступны для быстрого освоения их широким пользователем. Запрос и, если это необходимо, входные данные, описанные с использованием языковых средств данного пакета, определяют конкретную задачу, которую программная система должна решить. Управляющая программа системы анализирует полученный запрос и выбирает те обрабатывающие модули, которые необходимы для решения поставленной задачи. Обычно разные запросы требуют различных алгоритмов их реализации, а следовательно, различных совокупностей модулей.

Далее управляющая программа образует из выбранных модулей последовательность, называемую рабочей программой. Рабочая программа может создаваться как перед выполнением обработки данных (в случае реализации управляющей программы на языке управления заданиями), так и динамически — в ходе выполнения обработки данных.

1. 3. 2. Свойства программной системы нак формы программирования

Основным преимуществом данной формы программирования является высокая степень автоматизации процесса решения задач путем автоматического создания рабочей программы по запросу. Отметим, что число различных рабочих программ, которые могут быть созданы в процессе работы программной системы для реализации разных запросов, обычно достаточно велико. Таким образом, с помощью одной программной системы можно решать не одну задачу, а практически целый набор связанных между со-

* С позиций операционной системы шаг решения задачи представляется определенным загрузочным модулем, который в описании задания на языке управления заданиями выступает как шаг (или пункт) задания [17], [19].

бой прикладных задач. При этом от пользователя не требуется большого объема знаний для использования программной системы. Практически он лишь должен описать исходные данные и требуемый запрос на работу программы на языке программной системы. При такой простоте эксплуатации обеспечивается возможность

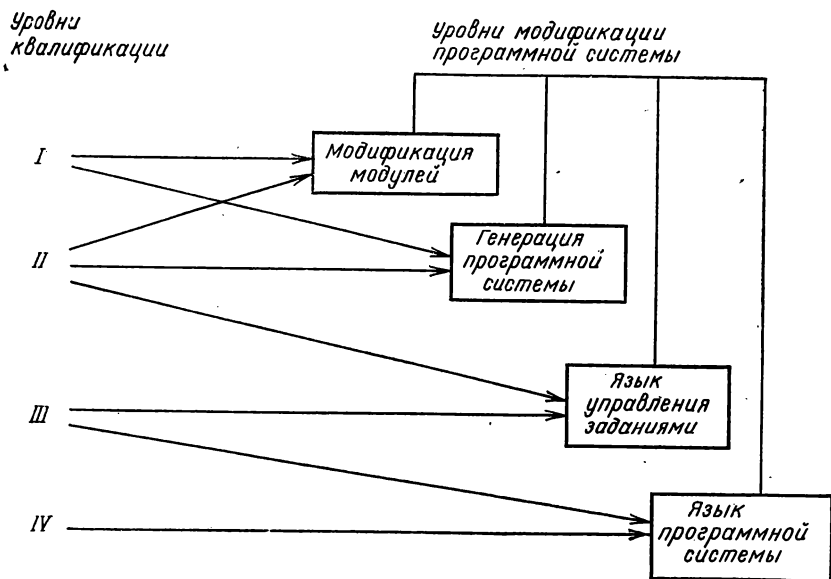


Рис. 11. Взаимосвязь «человек — программа» в процессе работы с программной системой:

I — разработчик программ; II — программист-пользователь; III — квалифицированный пользователь; IV — широкий пользователь

применения широким пользователем большинства программных систем.

Другое важное преимущество данной формы программирования заключается в ее гибкости, т. е. возможности внесения изменений несколькими способами без существенных затрат времени и средств (рис. 11).

Некоторые наиболее сложные и крупные программные системы предполагают перед началом эксплуатации в качестве необходимого этапа процесс генерации, подобный генерации операционных систем ЕС ЭВМ. В ходе генерации происходит настройка пакета на конкретную конфигурацию комплекса технических средств вычислительной системы пользователя, а также его привязка к специфике задачи пользователя, решаемой данной программой. При этом, как и в случае работы с операционными системами, пользователь получает возможность сформировать наиболее эффективную программу с позиций своих требований к решению задачи.

Используя язык управления заданиями операционной системы [17], [19], программист непосредственно перед выполнением программы может обеспечить реализацию определенных дополнительных требований по настройке пакета на конфигурацию технических средств или подключить к выполнению какие-либо специальные программы, расширяющие возможности данного пакета.

Языковые средства системы обеспечивают пользователю возможность определить свои требования к решению задачи. Таким образом, например, можно определить те отчеты, которые необходимо сформировать в результате выполнения программы, задать форматы выходных отчетов и т.п.

При необходимости более глубоких изменений и модификаций имеется возможность заменить один или несколько программных модулей на различных уровнях иерархии либо ввести в программную систему свои собственные модули. Подобные модификации почти не оказывают влияния на остальную часть программной системы благодаря развитой модульности и относительной независимости отдельных модулей в рамках программной системы. Это свойство данной архитектуры существенно облегчает процесс внесения изменений. Модификации обычно должны осуществляться разработчиками пакета или специалистами по сопровождению и лишь в исключительных случаях их может выполнить программист-пользователь, если он имеет соответствующую документацию и тексты программ пакета на исходном языке программирования.

Создание программных систем представляет сложный и трудоемкий процесс. Поэтому разработка программ такой структуры оправдана обычно в следующих случаях:

если сам процесс решения задачи должен по каким-либо причинам протекать автономно, без участия человека;

если требуется применение вычислительной системы пользователями, не являющимися специалистами в области программирования.

Программы, имеющие архитектуру программной системы, могут быть как методо-, так и проблемно-ориентированными. При этом разработку методо-ориентированных пакетов прикладных программ в форме программных систем целесообразно ограничить теми случаями, когда реализация одного метода или некоторой совокупности связанных между собой методов достаточно сложна в какой-либо другой форме или имеются некоторые дополнительные требования, которые не могут быть обеспечены программами другой архитектуры. Примером методо-ориентированного пакета прикладных программ, имеющего архитектуру программной системы, может служить программа для решения задач линейного программирования. В этом случае реализация программы в форме программной системы обусловлена тем, что методы линейного программирования широко распространены в различных сферах применения. Такую программу можно отнести к так называемым программам общего назначения. Ее могут применять пользователи различных категорий. При этом широкий пользователь с большим

успехом будет применять такую программу, реализованную именно в форме программной системы.

Проблемно-ориентированные пакеты прикладных программ обычно реализуются в форме программных систем. Это объясняется рядом причин.

Во-первых, проблемная ориентация в отличие от ориентации на метод создает возможность разработки прикладной программы в форме, не требующей от пользователя дополнительного программирования при решении прикладных задач. Иначе говоря, высокая степень автоматизации вычислительного процесса может быть обеспечена в основном проблемно-ориентированными программами. Из рассмотренных нами форм программирования только программные системы обеспечивают реализацию прикладных программ, обладающих этим свойством.

Во-вторых, проблемно-ориентированные пакеты прикладных программ предназначаются в основном для тех сфер применения, где основную роль играет широкий пользователь, для которого высокая степень автоматизации вычислительного процесса является важнейшим условием эффективной эксплуатации программы.

Таким образом, характеристики сфер применения и характеристики пакетов прикладных программ логически связаны с определенными формами программирования. Эта связь проявляется как тенденция в процессе разработки и эксплуатации прикладных программ.

1. 3. 3. Соотношение различных форм программирования

Архитектура программных систем имеет много общего и с библиотекой программ и с системой программирования, использующей специализированный язык. В то же время она имеет много особенностей.

С позиций структуры программная система, как библиотека программ и система программирования, использующая специализированный язык, представляет собой совокупность программ — модулей. Однако отдельные модули программной системы, как и модули системы программирования, могут не иметь самостоятельного значения, а предназначаться для реализации определенных частей единого алгоритма. Планомерное разделение алгоритмов решения задачи на отдельные модули — важный этап разработки программной системы, необходимый для обеспечения гибкости программы. В частности, поскольку отдельные модули программной системы могут быть в процессе решения задачи объединены различными способами и в различных сочетаниях, разработка программных систем требует создания стандартной системы сопряжения модулей в рабочей программе. При создании библиотеки обеспечение возможности стандартного сопряжения программ не представляет специальной задачи, поскольку здесь увязка отдельных программ в процессе решения прикладных задач возлагается на пользователя. Обычно поэтому программы библиотеки удовлетво-

ряют лишь соглашениям о связях, принятым в операционных системах ЕС ЭВМ [17], [19]. При разработке же программных систем этих соглашений зачастую оказывается недостаточно.

В плане логической обособленности и отсутствия для пользователя конкретного содержания модули программной системы сходны с программами, реализующими систему программирования с использованием специализированного языка. Можно отметить, что в том и в другом случае основой использования программы являются определенные языковые средства. Однако между специализированным языком программирования и языковыми средствами программной системы имеются существенные различия.

Специализированный язык программирования предназначен для описания алгоритма решения задачи пользователя. Модули системы программирования совместно с некоторыми обрабатываемыми программами операционной системы преобразуют программу, написанную на специализированном языке, в программу, выполняемую на вычислительной машине.

Языковые средства программной системы служат лишь для описания запроса к обработке и исходных данных, необходимых для решения задачи. Управляющая программа пакета организует обработку запроса, используя обрабатывающие модули программной системы. От пользователя не требуется описания алгоритма решения задачи.

В процессе развития форм и методов программирования могут появляться программы, имеющие архитектуры, отличные от рассмотренных выше. Некоторые из существующих и используемых в настоящее время программ включают элементы, присущие различным формам программирования. Так, в частности, типичная программная система для решения задач линейного программирования может быть объединена в пакет прикладных программ с системой программирования, обеспечивающей пользователя специализированным языком для формирования различных выходных документов по результатам решения указанных выше задач. Подобный пакет легко настраивается на специфику различных областей применения методов линейного программирования, которая обычно проявляется в необходимости компоновки результатов вычислений в определенной форме. В то же время сами широкоизвестные методы линейного программирования, реализованные в рамках программной системы, позволяют решать задачи из различных областей без специальной настройки. Использование специализированного языка и соответствующей системы программирования необходимо лишь на стадии привязки программы к специфике задач пользователя и в дальнейшем пользователь может, не прибегая к этим средствам, применять лишь программную систему, дополненную программами формирования необходимых отчетов.

Подобный сложный пакет прикладных программ обеспечивает преимущества двух различных форм программирования. Решение о разработке такого пакета может быть принято на основе анализа различных форм программирования, подобно приведенному.

выше. Вообще результатом изучения и формирования новых методов программирования является их целенаправленное применение при разработке прикладных программ, наилучшим образом удовлетворяющих требованиям эксплуатации.

1. 4. АРХИТЕКТУРА ПРИКЛАДНЫХ ПРОГРАММ И СФЕРЫ ИХ ПРИМЕНЕНИЯ

В процессе изучения и анализа прикладного программного обеспечения выработалось следующее деление пакетов прикладных программ по сферам применения и классам решаемых задач:

расширяющие возможности операционных систем;

общего назначения*;

для решения основных инженерных и научно-технических задач;

для решения задач экономического характера и задач АСУ;

для специальных применений.

В условиях постоянного расширения использования ЭВМ такое деление не позволяет четко отнести любые прикладные программы к определенным сферам применения, однако оно подчеркивает особое значение некоторых сфер применения, обусловленное широким распространением соответствующих классов прикладных задач в процессе эксплуатации современных ЭВМ. Рассмотрим подробнее специфику задач каждой группы.

1. 4. 1. Пакеты прикладных программ, расширяющие возможности операционных систем

Этот класс прикладных программ выделяется в связи с особенностями ЭВМ третьего поколения и предназначен для обеспечения некоторых режимов эксплуатации вычислительных систем или функционирования специальных технических средств вычислительных машин. Идентификация таких программ, как правило, не вызывает затруднений. Рассмотрим их особенности на примере программы, обеспечивающей использование ЭВМ в реальном масштабе времени [48] (более подробно режимы использования ЭВМ рассматриваются в гл. 5).

Работа с ЭВМ в реальном масштабе времени представляет собой один из режимов эксплуатации вычислительных машин, реализуемый на моделях ЕС ЭВМ [17]. Существует много определенных понятия «работа в реальном масштабе времени». Мы имеем в виду следующее.

Считается, что работа идет в реальном масштабе времени, если время ответа на запросы ограничивается внешними условиями решения задачи [48]. При этом подразумевается, что требуемое

* Класс прикладных задач общего назначения не представляет собой специальной сферы применения, однако рассматривается в связи с широким распространением этих задач у пользователей.

время ответа должно быть достаточно мало, поскольку в ином случае данное условие не предъявляет к вычислительной системе, реализованной на базе ЭВМ Единой системы, практически никаких специальных требований. Примерами использования вычислительных машин в реальном масштабе времени могут служить задачи управления технологическими процессами или задачи распределения и продажи билетов на аэровокзале.

При работе в подобном режиме необходимы дополнительные средства управления вычислительной системой, которые должны быть реализованы программно. Эти программы не включаются в состав основных операционных систем, так как они предназначены исключительно для организации данного режима и не требуются на большинстве вычислительных установок. Комплекс программ, реализующих дополнительные средства управления системой, используемой в реальном масштабе времени, выделяется в обособленный пакет прикладных программ, называемый монитором реального времени.

Поскольку в функции монитора (наряду с программами основной операционной системы) входит обычно управление работой многих различных пользователей или технических устройств, необходимым требованием является его независимость от каждого конкретного потребителя, т. е. требование автономной работы программы под управлением операционной системы. С учетом этого требования программа реализации монитора должна обеспечивать высокую степень автоматизации вычислительного процесса, поэтому монитор реального времени по архитектуре представляет собой программную систему.

Данная форма программирования широко применяется и для других пакетов прикладных программ, расширяющих возможности основных операционных систем.

1. 4. 2. Пакеты прикладных программ общего назначения

В процессе расширения области использования вычислительных машин выделяются прикладные задачи и математические методы их решения, которые являются общими для многих сфер применения ЭВМ. Например, методы теории массового обслуживания, многие статистические методы обработки данных (анализ временных рядов, корреляционный и регрессионный анализ и т. п.), методы математического программирования (линейное, параметрическое программирование и т. п.) используются в настоящее время в самых различных областях. Программные реализации подобных задач относятся к пакетам прикладных программ общего назначения.

В большинстве случаев эти программы представляют базу для решения сложных прикладных задач. Программы общего назначения обеспечивают вспомогательные средства для работы программистов-пользователей и квалифицированных пользователей,

являясь обычно методо-ориентированными. Однако некоторые общеприменимые математические методы используются для решения прикладных задач, имеющих большое самостоятельное значение. В этих случаях ориентация на метод может сочетаться с разработкой прикладной программы в форме программной системы с тем, чтобы обеспечить возможность решения задач широким пользователем и упростить работу пользователей других категорий. В качестве примера можно привести пакет прикладных программ, реализующий методы линейного программирования, широко распространенные как в инженерно-технических применениях, так и в экономике.

1. 4. 3. Пакеты прикладных программ для решения научно-технических задач

Основной особенностью инженерных и научно-технических применений является постоянное появление новых задач, требующих для своего решения и новых программ. В такой ситуации использование в качестве основного аппарата программных систем повлекло бы необходимость постоянной их модификации в связи с новыми задачами и применениями. Это привело бы к существенному повышению затрат на разработку и снижению эффективности эксплуатации прикладных программ. С другой стороны, квалификация инженерных и научных кадров в области применения ЭВМ в настоящее время в нашей стране такова, что для большинства специалистов применение в практике работы библиотек программ и специализированных языков программирования не представляет существенных затруднений. Более того, описанная выше специфика обуславливает использование программ такой архитектуры как основных, на базе которых пользователь может формировать программы для решения собственных задач. Использование методо-ориентированных пакетов прикладных программ, а следовательно, и указанных форм программирования, которые наилучшим образом соответствуют этому назначению, здесь полностью оправдано. Разработка программных систем в этой области может быть ограничена точно так же, как и для методо-ориентированных пакетов прикладных программ вообще.

Однако следует учитывать нечеткость самого определения сферы применения. Она может повлиять на правильность анализа. В некоторых случаях определенные группы традиционных научно-технических и инженерных задач могут выделяться и реализовываться в виде проблемно-ориентированных пакетов прикладных программ. Так, к инженерным задачам могут быть отнесены задачи расчета статике сооружений — группа логически связанных задач строительной механики. Их реализация в форме программных систем вполне возможна и даже удобна, поскольку имеющиеся в данной области прикладные задачи в достаточной степени отработаны, чтобы регулярно не претерпевать существенных измене-

ний. Подобную программу можно рассматривать как проблемно-ориентированную.

Отметим, что для программистов-пользователей и квалифицированных пользователей программирование прикладных задач в этой сфере применения существенно облегчается благодаря возможностям использования проблемно-ориентированных языков высокого уровня, предназначенных именно для научно-технических применений (Алгола, Фортрана и др.).

1. 5. НЕКОТОРЫЕ ОСОБЕННОСТИ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

В настоящее время, когда в нашей стране широкий размах приобретают работы по созданию и внедрению в практику управления предприятиями и организациями автоматизированных систем управления, одной из наиболее важных становится задача анализа и оценки эффективности разработки и эксплуатации прикладных программ АСУ, затраты на которые достаточно велики [55], [56], [60].

1. 5. 1. Типизация прикладных программ АСУ. Программная совместимость

Одним из основных путей снижения затрат на создание АСУ является типизация ее прикладного программного обеспечения, разработка которого представляет существенную долю общих затрат на создание автоматизированных систем управления.

Если учесть, что подобные затраты несет практически каждое предприятие, разрабатывающее автоматизированную систему управления, становится очевидным, что требование типизации прикладного программного обеспечения АСУ является основным в процессе его разработки. При этом принципиальная возможность типизации прикладных программ АСУ обеспечивается определенным единством форм и методов управления предприятиями и организациями, которое определяется существующими инструктивными и методическими материалами.

В этом плане характерен опыт разработки прикладных программ для решения задач управления за рубежом. Несмотря на различие организационных схем и систем управления различными фирмами, многие из них для целей управления используют общее прикладное программное обеспечение. Более того, принудительный ритм, задаваемый определенными программными решениями, сам способствует унификации систем управления. Отказ от признанных комплексов прикладных программ должен быть в каждом конкретном случае убедительно обоснован, поскольку он связан с существенным увеличением затрат на создание систем обработки данных. В качестве примера подобного комплекса программных средств для нужд управления промышленным пред-

приятием можно привести систему пакетов прикладных программ, объединенных в единый комплекс под общим названием PICS (The Production Information and Control System), разработанный фирмой IBM*.

Требование типизации прикладного программного обеспечения в сфере управления определяет одну из наиболее существенных особенностей этого класса прикладных задач, поскольку оно не проявляется с такой остротой практически ни в одной другой сфере применения ЭВМ. С другой стороны, при реализации типового программного обеспечения в автоматизированных системах управления предъявляются определенные требования как к техническим средствам ЭВМ, так и к форме программирования задач обработки данных. Поскольку данная сфера применения представляется одной из наиболее важных ввиду массового характера стоящих здесь задач, рассмотрим более подробно основные проблемы, связанные с реализацией прикладных программ для систем управления. При этом будем учитывать опыт, накопленный при разработке программного обеспечения АСУ на базе ЭВМ второго поколения.

Программная совместимость — предпосылка типизации. Анализ предпосылок и возможностей типизации программного обеспечения необходимо начать с рассмотрения важнейшего системного свойства группы вычислительных машин — программной совместимости. В предшествующие 8—10 лет широкое распространение в нашей стране получили электронные вычислительные машины второго поколения. Многие из разработанных к настоящему моменту АСУ базируются на этих ЭВМ. Например, ЭВМ типа «Урал», «Минск», БЭСМ, М-220 и другие. Основные технические характеристики этих машин находятся в довольно узких диапазонах, что позволяет говорить о примерно одинаковых их возможностях (за исключением отдельных машин, например «Минск-32» и БЭСМ-6). Системное программное обеспечение этих ЭВМ в основном состоит из трансляторов с различных проблемно-ориентированных языков программирования, всевозможных программ отладки и программ, предназначенных для обслуживания операций ввода-вывода.

Необходимо отметить, что эти вычислительные машины, мало отличаясь друг от друга по своим возможностям и техническим характеристикам, становятся принципиально различными при необходимости использования программ, разработанных для ЭВМ одного типа, на ЭВМ другого типа. Это свойство мы будем называть несовместимостью ЭВМ различных типов, или программной несовместимостью. Программная несовместимость с внешней логической точки зрения объясняется различием в системе команд и форматах представления команд и данных для различных вычислительных машин. Другой существенный недостаток рассмат-

* См.: E20-0280-2. The Production Information and Control System, Third Edition, IBM Corporation, 1968.

риваемых средств вычислительной техники состоит в отсутствии единой номенклатуры периферийных устройств, что также затрудняет обмен программной продукцией между пользователями.

Эти особенности ЭВМ второго поколения, безусловно, препятствуют типизации программного обеспечения АСУ. Практически в данной ситуации имеет смысл говорить о создании типовых программных решений лишь в рамках кооперации пользователей, имеющих ЭВМ одного типа. При этом отметим, что ориентация всех или большинства предприятий, разрабатывающих собственные автоматизированные системы управления, на ЭВМ какого-либо одного типа в условиях серийного выпуска ЭВМ многих различных типов невозможна по экономическим и техническим соображениям. Понятно, что существенные различия в размерах предприятий и сложности задач управления влекут, как правило, и различие в требованиях к мощности вычислительной системы. Один из возможных путей создания вычислительных систем различной мощности на базе ЭВМ одного типа состоит в создании многопроцессорных и многомашинных систем. Однако такой путь не является достаточно гибким. Кроме того, создание многопроцессорных систем с общим полем основной или внешней памяти связано для ЭВМ второго поколения с дополнительными техническими трудностями.

Переходя к оценке возможностей ЭВМ третьего поколения в плане типизации программного обеспечения, необходимо отметить наличие в составе ЕС ЭВМ ряда программно совместимых вычислительных машин. Технические характеристики машин этого семейства образуют широкий спектр и, следовательно, пользователь может выбрать ЭВМ с такими параметрами, которые наилучшим образом удовлетворяют его требованиям. Все модели программно совместимы на уровне программирования в машинных кодах, а следовательно, совместимы и на уровне машинно-ориентированного языка Ассемблера [4].

Одним из основных свойств ЕС ЭВМ является также полная техническая совместимость в плане единства номенклатуры внешних устройств. В связи с этим пользователь может увеличивать вычислительную мощность системы многими различными способами и, в частности, путем расширения состава периферийного оборудования или замены процессора более производительным. Отметим также, что в ЕС ЭВМ предусмотрена возможность объединения нескольких ЭВМ в многопроцессорную систему с общим полем основной или внешней памяти. Таким образом, проблема достижения совместимости для различных ЭВМ, как основы для реализации прикладных программ АСУ, находится гораздо ближе к своему решению для вычислительной техники третьего поколения.

Особенности программной совместимости для ЕС ЭВМ. В современных ЭВМ, оснащенных развитыми операционными системами, предполагается несколько иное построение программ пользователей, чем для вычислительных машин предшествующих

поколений. Если раньше программа состояла из команд, воспринимаемых к выполнению непосредственно аппаратными средствами вычислительной машины, то программа для машин типа ЕС ЭВМ дополнительно содержит обращение к операционной системе, которое необходимо для обслуживания данной программы (постановке ее в очередь заданий, выделения места в оперативной памяти и других операций). Без этого не может быть выполнена ни одна программа пользователя.

Следует также иметь в виду, что программы операционной системы, как и любые другие программы, выполняемые ЭВМ, требуют для своей работы определенных ресурсов вычислительной системы. Важнейшими среди них являются время работы процессора и занимаемый программами операционной системы объем оперативной памяти. Естественно, что чем больший объем управляющих и обслуживающих функций выполняется на вычислительной машине программами операционной системы, тем больше ресурсов требуют они для своей работы и, следовательно, меньше ресурсов остается для выполнения прикладных программ пользователей.

В связи с тем, что младшие модели вычислительных машин ЕС ЭВМ (ЕС-1020, ЕС-1030) являются менее мощными, экономически целесообразно, чтобы возможности их операционных систем были несколько ниже возможностей операционных систем для старших моделей (ЕС-1040, ЕС-1050). В результате этого появилась необходимость в разработке двух различных по возможностям операционных систем: ДОС ЕС — для младших моделей ЕС ЭВМ [19] и ОС ЕС — для старших [17]. Выполняя в принципе схожие функции, но имея различные возможности, эти операционные системы отличаются друг от друга языком обращения к ним (языком управления заданиями). Поэтому вторая составная часть любой прикладной программы — обращение к операционной системе на языке управления заданиями — при работе с ДОС ЕС выглядит иначе, чем при работе с ОС ЕС. Этими же причинами объясняется и различие в языке макрокоманд Ассемблера для ДОС ЕС и ОС ЕС [17], [19].

Таким образом, имея в виду особенности операционных систем, нельзя говорить о совершенной совместимости всех ЭВМ Единой системы в любых системах обработки данных. Эта совместимость зависит от типа операционной системы, управляющей данной конкретной вычислительной машиной. Однако следует иметь в виду, что никакой «внутренней» аппаратной ориентации на определенную операционную систему ЭВМ Единой системы не имеют [4]. У пользователя всегда остается возможность генерации на своей вычислительной машине той операционной системы, которая ему необходима в целях кооперации с некоторой группой пользователей и реализации необходимых ему прикладных программ с определенными возможностями.

Язык высокого уровня как средство для обеспечения совместимости. Существует принципиальная возможность преодоления программной несовместимости ЭВМ второго поколения путем исполь-

зования проблемно-ориентированных языков программирования: Алгола, Кобола, Фортрана и др. Эти языки создают благоприятные условия для обеспечения совместимости ЭВМ на более высоком уровне. Например, программа, написанная на Коболе, может быть обработана (с небольшими изменениями и ограничениями, обусловленными особенностями трансляторов различных типов ЭВМ) транслятором с языка Кобол, имеющимся для данной ЭВМ (практически для ЭВМ всех перечисленных выше типов разработаны такие трансляторы), и результатом обработки будет программа в машинных кодах, реализующая решение данной задачи на данной вычислительной машине. Однако следует постоянно иметь в виду, что при программировании на алгоритмических языках существенно увеличиваются затраты на эксплуатацию ЭВМ, которые, являясь постоянными, оказываются в конечном счете даже более существенными, чем однократные затраты на разработку. Это объясняется следующими причинами.

Во-первых, многие современные трансляторы с языков высокого уровня создают программы, требующие обычно существенно больше времени для своей работы по сравнению с программами, разработанными на машинно-ориентированном языке. Кроме того, эти трансляторы не обеспечивают эффективного распределения оперативной памяти, что также существенно влияет на общую эффективность применения языков программирования. Таким образом, затраты машинного времени, необходимого для эксплуатации прикладной программы, написанной на проблемно-ориентированном языке, увеличиваются в несколько раз по сравнению с программой, разработанной на машинно-ориентированном языке, что существенно повышает общие затраты на эксплуатацию автоматизированной системы управления.

Во-вторых, необходимо отметить, что большинство существующих алгоритмических языков программирования эффективно применимы только при решении задач определенного класса. Так, языки Алгол и Фортран применимы при решении научных и инженерных задач, в то время как язык Кобол более приспособлен для решения экономических задач. Однако в АСУ часто встречаются такие задачи, в которых достаточно сложные математические и логические расчеты необходимо проводить наряду с выдачей различного рода экономических таблиц и отчетов. Программирование таких задач на одном каком-либо языке еще более снижает эффективность получаемой программы.

Известны два подхода к решению подобных задач. Первый состоит в разбиении всего алгоритма решения задачи на крупные блоки и программировании этих блоков с использованием различных алгоритмических языков. Однако проблема организации связи между блоками в этом случае практически каждый раз представляет трудоемкую работу, которую программист вынужден выполнять самостоятельно. Вторая возможность состоит в создании универсального языка программирования, обеспечивающего эффективную программную реализацию обеих частей решения слож-

ной экономической задачи. Примерами таких языков являются ПЛ/1 и Алгол-68 [29], [32], [35]. Однако большой объем требуемых ресурсов вычислительной системы и сложность реализации трансляторов с этих языков не позволяют ввести эти средства в состав системного программного обеспечения ЭВМ второго поколения.

Таким образом, использование алгоритмических языков программирования на ЭВМ второго поколения, облегчая труд программистов, не обеспечивает существенного снижения суммарных затрат на эксплуатацию прикладного программного обеспечения АСУ. Сказанное, конечно, не означает, что применение языков программирования вообще не оправдано экономически. Во многих случаях при эксплуатации автоматизированных систем управления возникает потребность в программировании задач обработки данных, не предусмотренных разработанной системой, или программ однократного использования. В таких случаях проблемно-ориентированные языки представляют эффективные средства для оперативной разработки программ. Ориентация же только на эти языки при разработке крупных базовых прикладных программ АСУ типа программ, реализующих задачи функциональных подсистем, вряд ли оправдана.

Вообще использование языков высокого уровня более эффективно в тех сферах применения, где преобладают прикладные программы однократного использования и важнейшим критерием разработки программ является быстрота разработки и отладки, а не эффективность получаемой программы. В случае, когда основу использования ЭВМ составляют программы регулярного применения, относительная важность этих критериев смещается в сторону повышения эффективности прикладной программы и системы обработки данных в целом. При разработке программ регулярного применения в большинстве случаев эффективно применение машинно-ориентированных языков.

Общая тенденция в оценке языков высокого уровня в определенной мере меняется для вычислительных машин Единой системы, поскольку более высокие возможности этих ЭВМ позволяют разрешить некоторые из наиболее остро стоящих проблем в области использования языков программирования.

В рамках ЕС ЭВМ могут разрабатываться так называемые оптимизирующие трансляторы для различных языков программирования. Эти трансляторы, используя максимум информации о транслируемой прикладной программе, создают программы, по своей эффективности приближающиеся к программам, разработанным на машинно-ориентированных языках. Ограниченные возможности создания подобных трансляторов для ЭВМ второго поколения обусловлены высокими требованиями этих программ к вычислительным средствам машин. Так, для входящего в состав ЕС ЭВМ оптимизирующего транслятора с языка Фортран необходимо для работы 256К байт оперативной памяти [17] — больше, чем может обеспечить большинство вычислительных машин второго поколения.

В состав обеих операционных систем ЕС ЭВМ входит программа, называемая редактором связей [17], [19], позволяющая объединять отдельные модули, разработанные на различных языках программирования, в единую программу. Такая возможность, безусловно, упрощает процесс создания сложных программ.

Наконец, в рамках ЕС ЭВМ удастся реализовать трансляторы с универсальных языков программирования. В частности, обе операционные системы содержат трансляторы с языка ПЛ/1 [17], [19].

Все перечисленные средства создают предпосылки для более широкого применения в рамках ЕС ЭВМ языков программирования при разработке прикладных программ. Имея большой спектр возможностей, программист должен в каждом конкретном случае определять, какие языки и для каких этапов решения задачи удобнее использовать. Однако в некоторых случаях, когда эффективность получаемой программы существенно влияет на качество системы обработки данных в целом, действуют общие тенденции, сформулированные выше.

1. 5. 2. Типизация прикладных программ АСУ. Гибкость программных реализаций

Поясним на примере то содержание, которое мы вкладываем в понятие «типовое программное обеспечение».

Предположим, что для некоторого предприятия разработана качественная программа, реализующая на ЭВМ решение достаточно распространенной задачи управления производством. Вопрос состоит в том, может ли эта программа быть использована другим предприятием в АСУ без каких-либо изменений при условии, что это предприятие имеет ЭВМ того же типа. Можно утверждать, что как бы ни было высоко качество разработанной программы, изменения в том или ином объеме все же придется вносить. Это обусловлено специфичностью сложившихся традиционных систем управления в различных отраслях и на предприятиях, которая определяется целым рядом факторов и при создании АСУ может быть преодолена только до некоторой степени.

Таким образом, типовой прикладной программой следует называть такую программу, которая допускает внесение изменений без существенных затрат времени и средств, т. е. типовая программа должна быть легко адаптируемой к особенностям конкретного предприятия. Это свойство программы можно назвать ее гибкостью (или адаптируемостью).

Адаптируемые прикладные программы. Анализируя возможность создания гибких прикладных программ для ЭВМ второго поколения, необходимо отметить основную особенность программирования на этих вычислительных машинах — «монолитность». Как правило, здесь прикладная программа разрабатывается в виде последовательности команд, написанных одна за другой от первой до последней. Если программа и разделена на блоки, то каждый из них представляет собой также «монолитную» программу.

В каком соотношении находится проблема типизации программного обеспечения АСУ и «монолитная» форма программирования для ЭВМ второго поколения? Опыт показывает, что использование «монолитных» прикладных программ в качестве типовых неэффективно. Внесение каких-либо изменений, даже незначительных, в «монолитную» программу связано с большими затратами. Если же требуются серьезные изменения и модификации, затраты на их проведение становятся сравнимыми с затратами на разработку самой программы. Кроме того, оперативное внесение подобных изменений при «монолитной» форме программирования также затруднительно, так как обычно требуется переделка большей части программы. Таким образом, основная и наиболее распространенная форма разработки программ для ЭВМ второго поколения существенно усложняет решение задачи типизации системы прикладного программного обеспечения АСУ даже в рамках группы пользователей, эксплуатирующих ЭВМ одного типа.

Может возникнуть естественный вопрос: если существует специальная форма программирования — «монолитная» программа, то почему она не рассмотрена при описании архитектур прикладных программ? Это обусловлено двумя причинами.

Во-первых, были рассмотрены только регулярно используемые прикладные программы, в то время как существует сложность применения «монолитных» программ за пределами системы обработки данных той организации, для которой они создавались.

Во-вторых, в приведенной классификации «монолитные» программы нашли определенное место как элементы библиотек прикладных программ, где каждая отдельная программа предназначена для решения определенной простой задачи. При усложнении самой задачи и соответственном увеличении объема программы такая форма программирования уже не может считаться приемлемой.

Переходя к оценке возможностей типизации программного обеспечения на базе ЕС ЭВМ с точки зрения гибкости программирования, необходимо отметить возможность использования программных систем в качестве одной из основных форм программирования задач обработки данных в этой сфере применения (рис. 12). Подобная форма программирования существенно отличается от «монолитной» формы. Основным преимуществом программной системы является ее гибкость — возможность привязки к задаче пользователя несколькими способами (рис. 11).

Простота использования прикладных программ АСУ. Наряду с необходимостью типизации прикладного программного обеспечения АСУ существует и другое, не менее важное требование, вытекающее из самой специфики задач автоматизации управления. Оно состоит в необходимости использования вычислительных систем руководителями различных уровней на предприятиях и в организациях, т. е. в необходимости эксплуатации ЭВМ широким пользователем. Такой подход к решению задач автоматизации управления позволит, с одной стороны, снизить затраты на обслуживающий

персонал автоматизированных систем путем сокращения числа специалистов, составляющих промежуточное звено между руководящим персоналом и системой обработки данных. С другой стороны, простота эксплуатации систем обработки данных поможет преодолеть так называемый психологический барьер, существующий в отношении сложности использования автоматизированных систем управления.

Заметим, что речь здесь идет не о повышении квалификации руководителя до уровня специалиста по обработке данных, а о

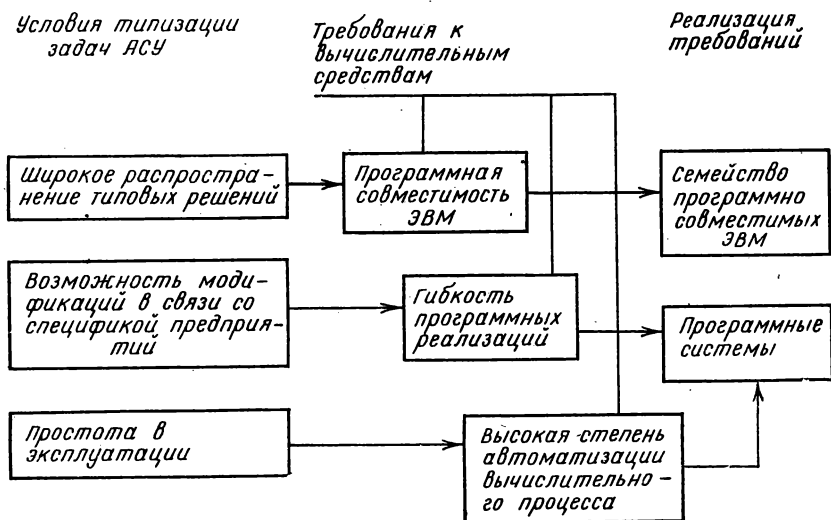


Рис. 12. Некоторые аспекты типизации задач АСУ

приближении вычислительных средств к нему, упрощении их применения. Подобное условие может быть реализовано прикладными программами с высокой степенью автоматизации вычислительного процесса, например программами, имеющими архитектуру программной системы.

Возможности реализации программных систем. В связи с изложенным возникает вопрос о возможности использования этой формы программирования на ЭВМ второго поколения. Действительно, при работе с вычислительными средствами второго поколения существует принципиальная возможность разработки программных систем как основы для типизации всей системы прикладного программного обеспечения АСУ. Более того, программирование обрабатывающих модулей системы является здесь традиционным и не представляет сколько-нибудь серьезных затруднений. Однако разработка и программная реализация управляющей части системы представляет собой сложную и трудоемкую задачу, требующую привлечения большого числа квалифицированных специалистов. Даже в тех случаях, когда программная система име-

ет относительно простую структуру и управляющая программа для нее в ЕС ЭВМ реализуется средствами операционной системы, для ЭВМ второго поколения необходима специальная реализация подобной программы. В противном случае на оператора всякий раз возлагается дополнительная работа по организации вычислительного процесса, что исключает возможность оперативной обработки данных и доступа пользователя к вычислительной системе.

Кроме того, не следует забывать, что в силу программной несовместимости различных типов ЭВМ второго поколения работа по созданию программных систем должна дублироваться, что существенно увеличивает затраты на разработку программного обеспечения АСУ.

Для ЭВМ третьего поколения наличие мощной операционной системы упрощает разработку управляющей программы системы. Такая программа может использовать весь спектр возможностей, заложенных в операционной системе. Применение операционной системы при разработке сокращает затраты на создание программной системы, а программная совместимость позволяет без каких-либо изменений использовать ее на любой модели ЕС ЭВМ (при наличии необходимого объема оперативной памяти и некоторых других технических параметров конкретной ЭВМ). В разряд ограничений не включается, например, конфигурация вычислительной системы, если для работы пакета не требуется каких-либо специальных внешних устройств. Настройка системы на конкретную конфигурацию, которая осуществляется оперативно и с малыми затратами, призвана обеспечить совместимость программ на уровне используемых технических средств.

Особенности разработки прикладных программ для АСУ. Разработка прикладных программ для автоматизированных систем управления предприятиями и организациями имеет определенную специфику, обусловленную квалификацией пользователей, характером самих задач и способами применения результатов их решения. В связи с требованием типизации прикладных программ важнейшим является вопрос перестройки и увязки традиционной системы управления с реализацией некоторых задач и функций управления на ЭВМ. При этом специфика любой программной системы, предназначенной для решения задач управления, состоит в том, что она предполагает уже определенную организацию системы управления для своего использования. Это не означает, конечно, что программа всегда жестко привязывается к какой-либо четко определенной системе управления. Наоборот, она должна обладать гибкостью, о которой говорилось выше. Однако любая программа, предназначенная для решения задачи управления, предполагает определенный подход к ее решению. В этом смысле такая программа представляет не только инструмент управления, но и задает некоторый ритм в реализации данной функции управления (рис. 13). В связи с этим программная система для решения задач АСУ должна представлять собой результат совместной рабо-

ты разработчиков программного обеспечения и специалистов в области управления. Только в этом случае трудоемкая и дорогостоящая разработка подобной системы будет экономически оправдана. Более того, создание широкого класса программных систем для нужд АСУ может стимулировать необходимую работу по дальнейшему изучению и типизации функций управления с учетом специфики отдельных предприятий и организаций. Такое понимание целей и задач создания программного обеспечения для систем управления позволит существенно повысить эффективность разработа-

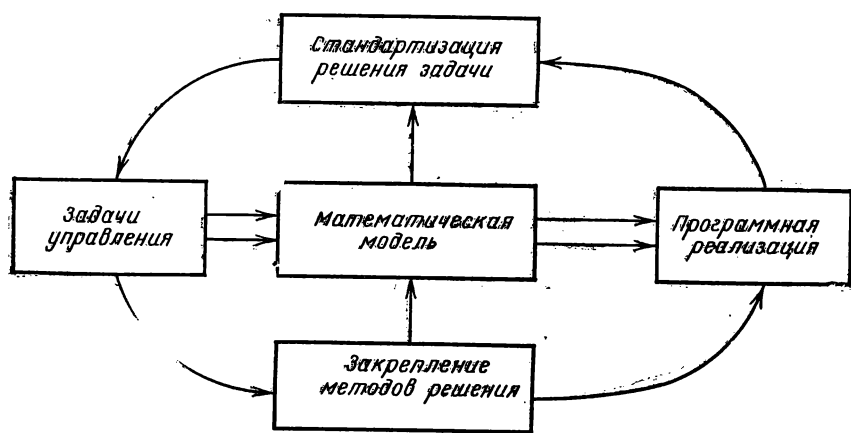


Рис. 13. Логика автоматизации управления

тываемых АСУ и поставить их разработку на индустриальную основу.

Целенаправленное решение задачи типизации функций управления на базе разрабатываемых программных систем требует включения в состав документации к ним материалов, представляющих общие рекомендации по организации решения соответствующих задач на предприятиях. При этом в таких рекомендациях должны затрагиваться лишь те стороны вопроса, решение которых регламентируется программой или же желательно для ее эффективного использования. Таким образом, гибкость рекомендаций будет соответствовать гибкости самой программной системы и предполагать возможность организации управления в соответствии со спецификой данного предприятия. Кроме того, в документации должны содержаться рекомендации по организации эксплуатации программной системы в общей системе обработки данных. Сюда относится реализация функций сбора данных, порядок их обновления, обработки и т. д.

Глава 2

ПРИКЛАДНЫЕ ПРОГРАММЫ В СИСТЕМЕ ОБРАБОТКИ ДАННЫХ

Анализ прикладных программ с позиций требований пользователей, учитывающий специфику сфер применения ЭВМ, позволяет выявить некоторые общие вопросы разработки и эксплуатации программы. В этой связи необходимо отметить, что целый ряд новых задач возникает в процессе организации интегрированных систем обработки данных. Важнейшей из них является обеспечение взаимосвязи отдельных прикладных программ в рамках интегрированной системы. От правильного решения этой задачи зависит эффективность системы обработки данных в целом. В этой главе будут рассмотрены специальные средства для организации программного обеспечения в интегрированных системах.

2. 1. НЕКОТОРЫЕ ПРОБЛЕМЫ СОЗДАНИЯ ИНТЕГРИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ДАННЫХ

Система обработки данных. Создание систем обработки данных (СОД) связано с выбором и организацией комплекса технических средств и разработкой прикладного программного обеспечения.

При выборе технических средств системы обработки данных пользователь ограничен той номенклатурой, которая имеется для данного типа ЭВМ. На первом этапе решения этой задачи уточняется состав и структура комплекса технических средств — его конфигурация. Конфигурация комплекса технических средств определяется на основе анализа объема и характера прикладных задач, для решения которых создается СОД. Широкий спектр технических средств ЕС ЭВМ обеспечивает возможность выбора эффективной конфигурации, хотя сама по себе эта задача достаточно сложна. При этом, как правило, не стоит вопрос о разработке новых технических средств для оснащения СОД.

Процесс создания программного обеспечения системы обработки данных происходит несколько иначе. Наряду с выбором уже имеющихся программных средств (таких, как операционная система, состав и структура которой определяются перед процессом генерации, а также ранее разработанные прикладные программы) создаются новые программы, предназначенные непосредственно для данной системы. В связи с этим разработка программного обеспечения является важнейшей частью работ по реализации системы обработки данных и по объему и сложности эти работы значительно превосходят ту, которая проводится при выборе конфигурации комплекса технических средств.

В процессе функционирования система обработки данных для пользователя может представляться «черным ящиком»: на вход которого подаются исходные данные, а на выходе получаются необходимые результаты. При этом данные могут вводиться в систему либо извне (с перфокарт, перфолент и др.), либо уже находиться в системе на магнитных носителях (магнитных лентах, дисках и др.). В большинстве случаев, если СОД находится в состоянии эксплуатации достаточно длительное время, большая часть данных хранится в системе.

Вопросы хранения, поиска и обработки данных приобретают в современных СОД первостепенное значение и оказывают существенное влияние на структуру и принципы функционирования СОД в целом. Особенно важно обеспечить эффективное решение этих вопросов в интегрированных системах обработки данных, т. е. в системах, предназначенных для реализации комплекса взаимосвязанных задач. В этом случае результаты решения одних задач могут в качестве исходных данных влиять на решение других и т. д. Подобные связи между задачами, естественно, находят отражение и в прикладных программах, реализующих их решение. Системы обработки данных, в которых отдельные прикладные программы связаны едиными исходными данными и результатами обработки, в настоящее время получают широкое распространение. Разработка таких систем представляет собой одну из важнейших задач применения вычислительной техники в особенности для автоматизированных систем управления. Ниже рассмотрены возникающие здесь вопросы и пути их решения.

2. 1. 1. Ранние концепции создания систем обработки данных в АСУ

Задачи управления предприятием или организацией тесно связаны между собой и образуют единый комплекс. Однако до последнего времени эта особенность систем управления в силу целого ряда объективных причин редко находила отражение в системах обработки данных для АСУ. Эти системы строились в виде набора обособленных прикладных программ, каждая из которых решала определенные задачи управления производством (например, задачи материально-технического снабжения, бухгалтерского

учета и т. п.). Исходные данные для решения этих задач объединялись в массивы*, используемые соответствующими программами. Различные программы обычно имели собственные массивы данных, с которыми они работали. Это обусловлено несколькими причинами.

При разработке новой программы, как правило, требуется новый массив. Состав реквизитов в записи этого массива может почти полностью совпадать с составом реквизитов какого-либо из уже имеющихся массивов и отличаться от него одним или несколькими элементами данных. Например, в массиве кадрового состава предприятия может содержаться реквизит должности сотрудников, но не иметься сведений об их заработной плате, необходимых для решения задач бухгалтерского учета и т. д. Расширение существующего массива путем введения в каждую запись нового реквизита является сложной и трудоемкой работой для программиста. Такая работа включает, с одной стороны, внесение изменений и дополнений в существующий набор данных, а с другой — изменение всех тех программ, которые ранее работали с этими данными. При этом программист должен детально знать организацию массива с точки зрения его логической структуры и размещения на внешних носителях информации (магнитных лентах, дисках и др.). Для переделки имеющихся программ также требуется детальное знание их внутренней реализации.

При укрупнении массива путем введения новых реквизитов время и сложность поиска данных увеличиваются. Соответственно увеличивается и объем той части программы, которая предназначена для поиска данных. Особенно существенно увеличение в случае, когда в одном и том же массиве необходимо вести поиск по различным элементам данных. Например, нас могут интересовать анкетные данные Иванова Дениса Федоровича или всех сотрудников старше 40 лет. В первом случае реквизитом, по которому ведется поиск, является фамилия, во втором — возраст.

Для решения различных прикладных задач и, следовательно, для разных программ могут требоваться разные связи между элементами данных в рамках одной записи или между записями в массиве. Например, некоторые задачи материально-технического снабжения на предприятии могут требовать массивов, содержащих сведения в форме «продукция — составляющие детали», а некоторые задачи оперативного управления производством — массивов в форме «детали — обрабатывающие станки». Попытка объединения таких массивов в общий массив повлечет существенное дублирование данных, поскольку различные изделия могут требовать одинаковых деталей (рис. 14, а).

* Массив (набор данных или файл) логически представляет собой поименованную совокупность записей (логических записей), описывающих однородные предметы [17]. Каждая запись описывает один предмет. Отдельные элементы записи, представляющие определенную характеристику предмета, называются элементами данных или реквизитами. Например, в массиве кадрового состава предприятия элементом данных является указание специальности, возраста, образования и т. п.

При проектировании прикладного программного обеспечения существует возможность организации программ и массивов таким образом, чтобы различные программы использовали одни и те же массивы. Однако в этом случае расширение системы обработки

данных вызывает еще более сложные проблемы модификации уже имеющегося задела программ и массивов данных, поскольку кроме изменения отдельных программ здесь необходимо учесть изменения связей между ними. Более того, в процессе работы нескольких программ с одним и тем же массивом данных каждая отдельная программа не использует, как правило, всех реквизитов записей. При этом существует возможность неправильного изменения неиспользуемых элементов данных при работе программ.

Все эти обстоятельства вызвали в определенный период необходимость построения систем обработки данных в форме «тандемов»: программа — массив данных для ее работы.

Эксплуатация подобных систем выявила ряд недостатков, наиболее существенным из которых является дублирование данных. Поскольку различные программы часто требуют одинаковых элементов данных, эти элементы неизбежно должны

дублироваться в соответствующих массивах (рис. 14, б). В крупных системах обработки данных дублирование может существенно повысить требования к объемам оперативной и внешней памяти вычислительной системы.

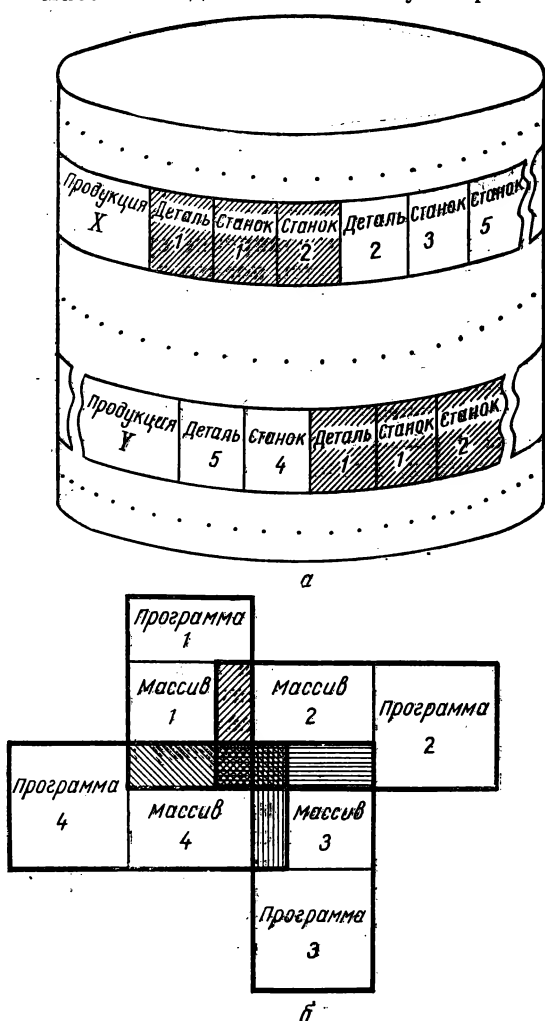


Рис. 14. Дублирование данных:
а — при укрупнении массивов; б — в системе обработки данных с тандемами

С другой стороны, системы обработки данных в АСУ характеризуются внутренней связью решаемых задач, которая должна проявляться и в программах. Подобная связь осуществляется путем использования результатов работы одних программ в качестве исходных данных для других. Поскольку число таких связей между задачами в системе управления велико, СОД с тандамами представляет сложную систему, требующую постоянного вмешательства обслуживающего персонала системы для упорядочения обмена дан-

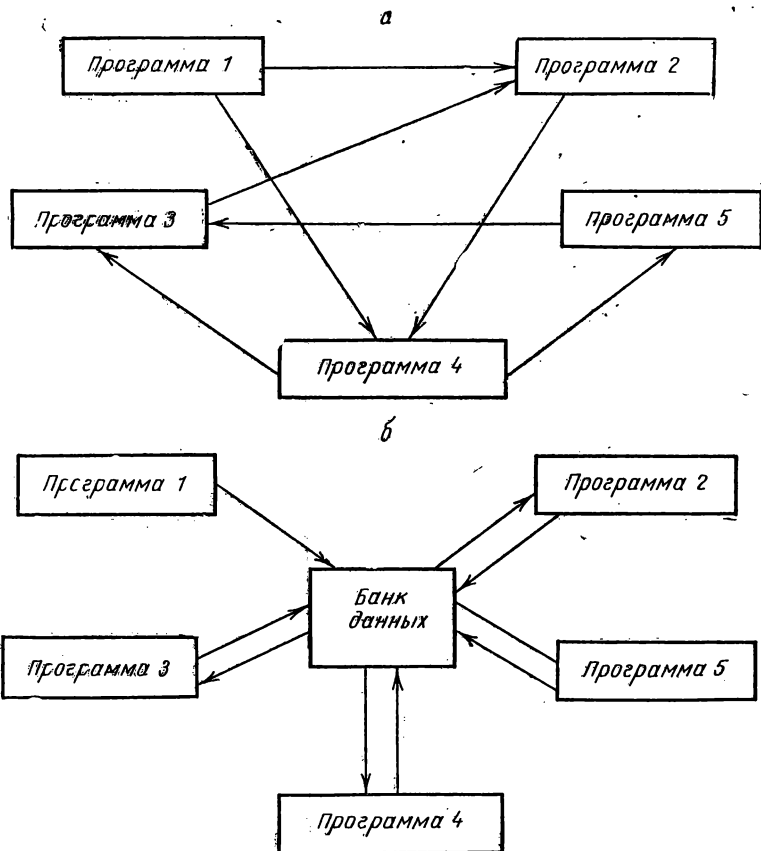


Рис. 15. Примерная схема связей между программами:
а — в СОД с тандамами; б — в СОД с банком данных

ными между программами (рис. 15, а). Такая организация СОД является трудоемкой в эксплуатации и ненадежной.

При функционировании системы обработки данных в АСУ одним из важнейших моментов является точная и своевременная корректировка (обновление) информации в массивах данных. Она обеспечивает два главных требования системы управления: оперативность получения информации и ее достоверность. При дублирова-

нии элементов данных в различных массивах необходимо обращать особое внимание на операции обновления данных, поскольку корректировки должны быть проведены во всех массивах, где данный реквизит встречается. При обновлении дублированных данных создаются предпосылки для появления ошибок и несоответствий в массивах СОД. Кроме того, утрачивается оперативность внесения изменений.

Таким образом, требования эксплуатации интегрированных систем обработки данных определяют необходимость специальных решений в процессе работы с данными. Эти решения должны обеспечить эффективность функционирования системы в целом.

2. 2. ЦЕНТРАЛИЗАЦИЯ ДАННЫХ В СИСТЕМЕ ОБРАБОТКИ ДАННЫХ ДЛЯ АСУ

2. 2. 1. Банк данных—центральное звено интегрированной системы обработки данных

Решение всего комплекса рассмотренных задач концентрируется в идее создания специальных программных средств, автоматизирующих работу с данными. С помощью этих средств должна осуществляться связь между программами пользователя и данными, имеющимися в системе. Комплекс массивов данных и программ, предназначенных непосредственно для организации централизованного хранения и поиска информации, а также для связи с прикладными программами, выполняющими ее обработку, мы называем банком данных.

В системе, основанной на банке данных, прикладные программы обращаются за данными для обработки не к внешним носителям информации, а к программам банка — системе управления базой данных, которая в свою очередь организует поиск, ввод и предоставление информации соответствующим программам из специально организованных массивов — базы данных (рис. 16). При таком способе организации работы с данными обычно говорят о логической форме представления данных для прикладных программ. Альтернативой этому построению системы обмена данными между прикладной программой и внешними устройствами является организация непосредственного обращения от программ, обрабатывающих данные, к внешним устройствам. В последнем случае говорят о физической форме представления данных, поскольку при таком обращении необходимо учитывать тип запоминающего устройства, хранящего информацию, принципы организации массивов и т. п.

Понятия логической и физической форм представления данных не являются абсолютными. В частности, операционные системы ЕС ЭВМ содержат набор средств, обеспечивающих логическую форму представления данных для прикладных программ [17], [19]. Однако эти средства не освобождают программиста полностью от таких функций, как организация данных, их поиск, выделение не-

обходимых элементов данных из записей и др. Программное обеспечение банка данных дает возможность существенно расширить функции операционных систем в плане реализации логической формы представления данных и упростить работу программистов. Тем самым создается основа для разработки интегрированных си-

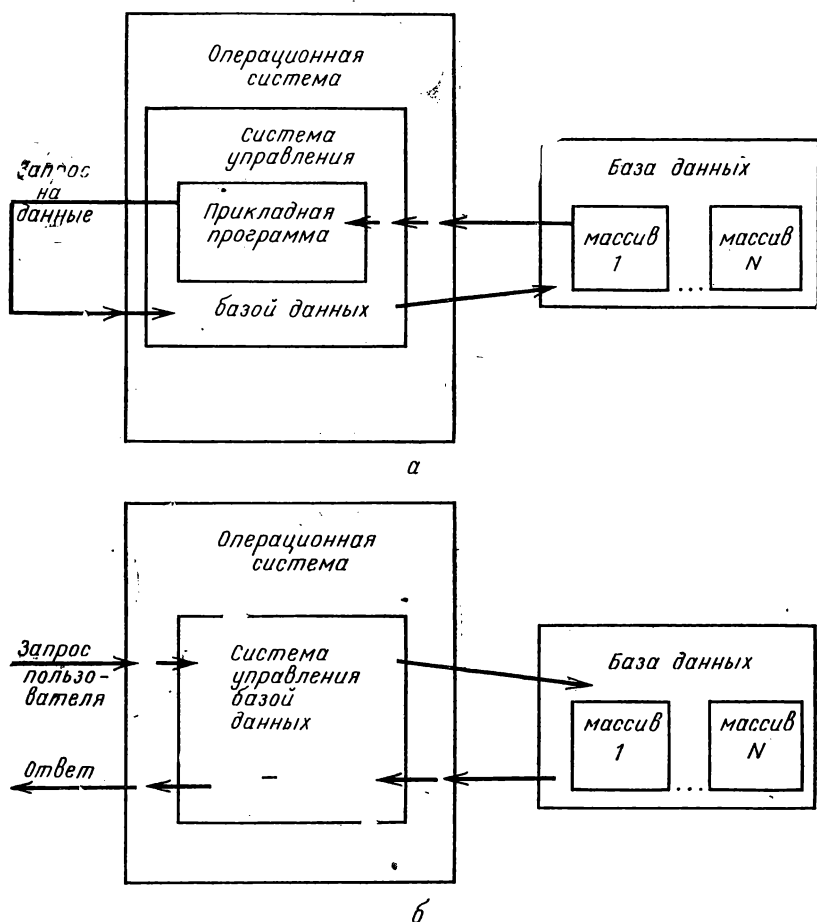


Рис. 16. Взаимодействие с системой управления базой данных:
а — обращение из прикладной программы; б — непосредственный запрос пользователя

стем обработки данных и обеспечивается возможность решения многих задач работы с данными. В частности, сложные связи между программами в СОД с тандамами заменяются более простыми связями между отдельными программами и системой управления базой данных (см. рис. 15, 16, а). С другой стороны у пользователя появляется возможность непосредственно получать сведения из

банка данных, не прибегая к их обработке с помощью прикладных программ (рис. 16, б).

Рассмотрим основные элементы, составляющие банк данных, принципы их построения и функционирования.

2. 2. 2. База данных

База данных представляет собой совокупность массивов, организованных таким образом, чтобы обеспечить хранение информации без дублирования, а также простой и оперативный поиск информации по запросу. В процессе построения базы данных вся

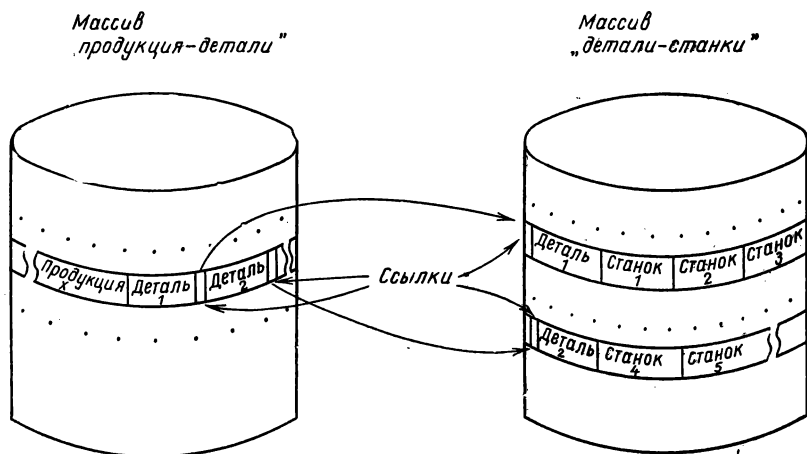


Рис. 17. Организация связи между массивами в базе данных

необходимая информация разбивается на массивы и посредством ссылок* устанавливается связь между отдельными элементами данных в рамках записи и между записями в массивах. Это обеспечивает существенное сокращение дублирования информации в базе данных. Например, можно организовать базу данных, состоящую из двух массивов: в первом содержатся записи вида «продукция — детали», во втором — «детали — станки». Снабдив в каждой записи первого массива каждый реквизит, характеризующий деталь, ссылкой на соответствующий элемент второго массива, мы получим возможность, обращаясь к массиву «продукция — деталь», пользоваться соответствующими данными из массива «детали — станки» (рис. 17).

Говоря о возможности работы с записями из различных массивов, следует иметь в виду, что она реализуется программами системы управления базой данных, входящими в банк. Организа-

* Ссылка — элемент данных или часть элемента данных записи, предназначенная для указания местонахождения другой записи этого или другого массива или другого элемента данных рассматриваемой записи.

ция связей между массивами с помощью прикладных программ обработки данных обусловила бы увеличение объема этих программ и дублирование той их части, которая выполняет функцию связи.

При разработке базы данных для интегрированной системы обработки данных разделение информации на массивы и организация связей между массивами должны проводиться в соответствии с требованиями программ обработки данных. Это обеспечит оперативность поиска информации в базе данных и передачи ее прикладной программе. В зависимости от возможностей программного обеспечения банка данных, которое будет применяться, пользователь может организовать в базе данных то или иное число массивов, объемы которых обычно ограничиваются лишь наличием определенного числа накопителей информации в рамках конкретной вычислительной системы. Точно так же допустимая сложность связей между массивами, обусловленная требованиями прикладных задач, зависит от выбранного программного обеспечения, обслуживающего базу данных.

2. 2. 3. Система управления базой данных

Система управления базой данных (СУБД) представляет собой комплекс взаимосвязанных программ, предназначенных для решения определенной задачи — обеспечения относительной независимости прикладных программ пользователя от конкретной организации массивов информации. Таким образом, подобная система организуется в виде пакета прикладных программ.

С точки зрения архитектуры СУБД представляет собой типичную программную систему, поскольку, работая со многими прикладными программами, она сама должна быть независимой от отдельных программ и, следовательно, должна обеспечить высокую степень автоматизации вычислительного процесса. Напомним, что из рассмотренных форм программирования наиболее высокую степень автоматизации обеспечивают именно программные системы.

Важнейшее свойство

СУБД состоит в их относительной независимости от конкретного содержания данных, образующих массивы базы данных. Практически, с учетом некоторых ограничений, определяемых конкретной реализацией, одна и та же система управления базой данных может быть использована для ведения совершенно различных массивов информации, предназначенных для разных применений. Подобная

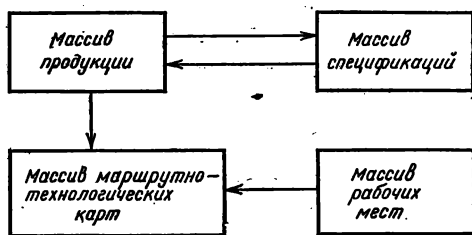


Рис. 18. Структура базы данных производственного предприятия для программ BOMP и MRP

независимость определяет гибкость системы управления базами данных и обеспечивает возможность стандартизации этих программ в рамках совокупности определенных типичных требований пользователей. Так, могут существовать системы управления базами данных, ориентированные на применение в АСУ предприятий с дискретным характером производства. Примерами таких систем могут служить пакеты прикладных программ Bill of Material Processor (BOMP) и Manufacturing Record Processor (MRP)*, разработанные фирмой IBM для вычислительных машин системы 360. Эти программы функционируют под управлением операционных систем DOS/360 и OS/360 [16], [23] соответственно, мало отличаясь по внутренним принципам работы и способам эксплуатации. Опыт использования этих программ показывает их гибкость для данной сферы применения, однако вне этой сферы возможности их ограничены. Программы BOMP и MRP предназначены для ведения базы данных, содержащей четыре массива. Реализация связей между этими массивами показана на рис. 18. Массив продукции хранит записи, описывающие каждый элемент продукции (не обязательно окончательный), изготавливаемый на предприятии. Адресные ссылки связывают этот массив с массивом спецификаций, где указывается применяемость данного элемента продукции при изготовлении более крупных элементов. Кроме того, массив продукции связан с массивом маршрутно-технологических карт, который служит для описания типов оборудования, необходимого для изготовления определенного элемента продукции (виды инструментов, станков и т. п.). Массив рабочих мест предназначен для описания всех типов оборудования, имеющегося на предприятии. Практически только два массива из четырех являются основными: массив продукции и массив рабочих мест. Они описывают материальные объекты: изготавливаемую продукцию и имеющееся оборудование. Два других массива служат для организации связей между отдельными записями массива продукции (массив спецификаций) и между массивом продукции и массивом рабочих мест (массив маршрутно-технологических карт). Совокупность перечисленных массивов с учетом связей между ними образует базу данных. Такая база данных позволяет детально описать процесс производства и проводить необходимые расчеты для решения целого ряда задач управления.

Системы управления базой данных BOMP и MRP содержат три крупных комплекса программ, обеспечивающих:

- подготовку и организацию вводимой информации;
- создание и обновление массивов;
- поиск необходимых данных в этих массивах.

На основе пакета BOMP разработан ряд программных систем, которые обеспечивают решение задач определенных функциональ-

* См.: GC 20-1619-8. Catalog of Programs for IBM System/360. Models 25 and above (January 1971), Ninth Edition, IBM Corporation, 1970. В дальнейшем при упоминании прикладных программ фирмы IBM специальных ссылок на этот документ не делается.

ных подсистем системы управления с использованием информации, сосредоточенной в базе данных. Комплекс этих программ получил общее название PICS.

Наряду с описанными существуют и универсальные системы управления базами данных, обслуживающие практически неограниченное число массивов с произвольными связями*. Однако здесь перед пользователями возникает ряд новых задач, без решения которых эксплуатация системы невозможна. Это разработка структуры базы данных, генерация системы управления базой данных в соответствии с требованиями конкретной области применения, разработка программ, реализующих решение задач пользователя с применением информации, сосредоточенной в базе данных, или привязка к системе уже существующих программ. Необходимость решения этих задач пользователем — негативная сторона универсальности подобных систем.

2. 2. 4. Языковые средства систем управления базой данных

Как и любая программная система, система управления базой данных имеет собственные языковые средства. С помощью этих средств решаются три важные задачи:

- описание баз данных;
- обеспечение доступа прикладных программ к базе данных;
- обеспечение доступа пользователей непосредственно к базе данных.

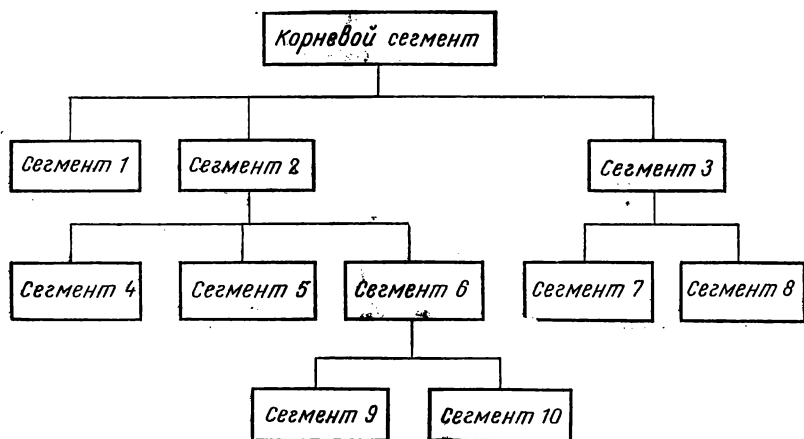
В зависимости от конкретной реализации системы управления базой данных языковые средства этой системы обеспечивают возможность в большей или меньшей степени реализовать каждую из этих задач.

Описание базы данных. Описание базы данных имеет в системах управления базами данных две основные цели: общее описание базы и описание базы для прикладной программы.

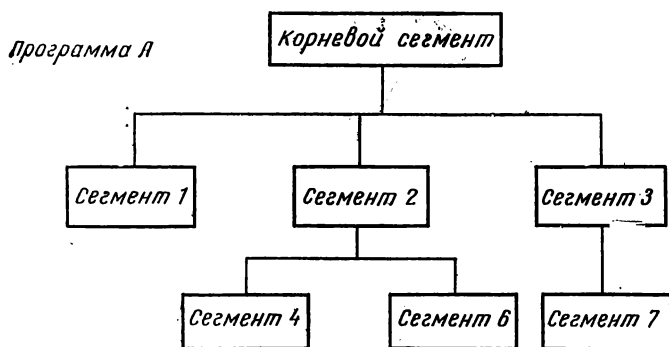
Общее описание базы данных представляет полную структуру записей базы в логической форме и необходимо для работы СУБД. Система управления базой данных обеспечивает преобразование записей базы, расположенных на внешних носителях информации, из физической формы представления в логическую. Это преобразование задается общим описанием базы данных.

Наиболее удобной логической формой представления записей является иерархическая структура, пример которой показан на рис. 19, а. Элементы иерархической структуры называются сегментами. С позиций построения записей в массивах сегменты соответствуют элементам данных (реквизитам). Иерархическая структура всегда содержит корневой сегмент записи. Остальные сегменты

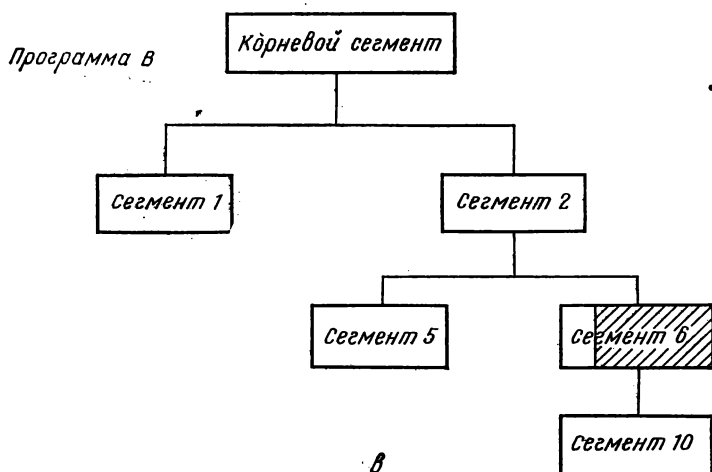
* См., например: Брусенков И. В. [и др.]. Программная система управления базами данных СИНБАД. — В сб.: Алгоритмы и организация решения экономических задач. Вып. 6. М., «Статистика», 1975.



а



б



в

Рис. 19. Логические структуры записей: а — общая; б и в — воспринимаемые различными программами

структуры называются зависимыми. Сегменты 1, 2 и 3 (см. рис. 19, а) являются зависимыми сегментами первого уровня, сегменты 4—8 — зависимые второго уровня и сегменты 9, 10 — зависимые третьего уровня. Сегмент, стоящий по иерархии непосредственно над зависимыми сегментами, называется родительским по отношению к ним. Так, сегмент 3 является родительским по отношению к сегментам 7 и 8; сегмент 2 — родительским по отношению к сегментам 4, 5 и 6 и т. д. В иерархической структуре каждый сегмент существует только в связи со своим родительским сегментом. Приведенная логическая структура позволяет достаточно просто реализовать передачу прикладной программе любой цепочки данных, начинающейся с корневого сегмента и определяемой связями, заданными в структуре. Например, возможны следующие последовательности сегментов: корневой — 1, корневой — 2—6—9, корневой — 3—8 и т. п.

Описание базы данных для прикладной программы также реализуется языковыми средствами системы управления базой. С помощью этих средств можно описать ту часть общей логической структуры, которая необходима для работы данной прикладной программы. На рис. 19,б и 19,в показано, как логическая структура, изображенная на рис. 19,а, воспринимается двумя различными программами с различным описанием базы данных для них. Программа А воспринимает логическую структуру записи, содержащей корневой сегмент, сегменты 1, 2 и 3 первого уровня и сегменты 4, 6 и 7 второго уровня. База данных, воспринимаемая программой В, состоит из корневого сегмента, сегментов 1, 2, 5, 6 и 10. При этом данная программа имеет ограниченный доступ к сегменту 6.

В этих примерах обе программы могут воспринимать корневой сегмент логической записи и сегменты 1 и 2 (сегмент 6 не может быть воспринят программой В полностью). За исключением этих сегментов записи в базе данных выглядят совершенно различно для программ А и В.

С помощью языковых средств определяется, какую часть полной логической структуры записи может воспринимать программа. В отношении сегмента 6 программе В разрешен доступ только к части сегмента, характеризующей его взаимосвязи с зависимыми сегментами. Этот сегмент содержит информацию, которая не должна быть доступна программе В, так как эта информация не требуется для ее работы. Однако программе В требуется доступ к информации, содержащейся в сегменте 10, для которого сегмент 6 является родительским. В связи с этим программе В требуется частичный доступ к сегменту 6.

Заметим, что при такой реализации базы данных и взаимодействия с ней прикладных программ любые изменения, например, в сегменте 5 коснутся лишь программы В. Модификации программы А при этом не потребуются, так как она не использует информацию, содержащуюся в сегменте 5.

Доступ прикладной программы к данным. Вторая важная функция языковых средств состоит в обеспечении доступа программиста к базе данных, которая не организована специально для его нужд. С помощью этих средств могут формулироваться запросы для получения прикладной программой тех или иных требуемых данных (элементов данных), имеющихся в базе. Подобный запрос может быть вставлен в текст программы, написанной на каком-либо из языков высокого уровня. Так, система СИНБАД [22] обеспечивает возможность использования языков Кобол, ПЛ/1 и Асемблера.

Система управления базой данных осуществляет поиск требуемой информации по запросу и выдает результаты поиска прикладной программе. При этом отношения логической взаимосвязи данных преобразуются в физические адреса данных на внешних носителях. Языковые средства избавляют программиста-пользователя от необходимости знать физическое местоположение его данных в базе. Кроме того, если при неизменном отношении логических взаимосвязей физическое размещение данных изменяется, не требуется модификаций прикладной программы.

Непосредственный доступ пользователя к данным. Система управления базами данных может обеспечивать непосредственную связь пользователя с банком данных. Такой способ эксплуатации банка позволяет оперативно получать необходимую информацию. При этом повышается эффективность применения банка и у пользователя появляется возможность непосредственного контакта с вычислительной системой.

Основная задача при реализации прямой связи между пользователем и банком данных состоит в обеспечении простых средств формулирования запроса. Эффективное использование банков данных в качестве справочных систем предполагает создание по возможности простых правил быстрого формулирования запроса, которые могут быть выполнены даже пользователем, не являющимся специалистом по программированию. Система может содержать специальные средства для описания пользовательских запросов. Необходимо отметить, что такие средства обычно весьма просты и не предназначены для формулирования сложных задач. Преимущества этих языковых средств: простота в употреблении, возможность быстрого формулирования запроса и наглядность. Усложнение языка отрицательно сказалось бы на оперативности и надежности системы, значительно снизив ее эффективность как справочной системы, хотя и позволило бы решать с ее помощью более сложные задачи.

Все рассмотренные функции языковых средств могут быть в той или иной степени присущи любой конкретной СУБД. При этом описание базы данных выполняет в системе контролирующие функции, а доступ к базе от прикладных программ или непосредственно от пользователей обеспечивает необходимую работу с данными, хранящимися в базе данных.

2. 2. 5. Преимущества применения банков данных в системах обработки данных

Важнейшим преимуществом, которое обеспечивается банком данных, является независимость данных. Это свойство организации данных состоит в том, что пользователи могут представлять лишь логическую структуру данных, не интересуясь их физическим расположением на внешних носителях. Преобразование физической структуры в логическую, задаваемое описанием базы данных, является функцией системы управления базой.

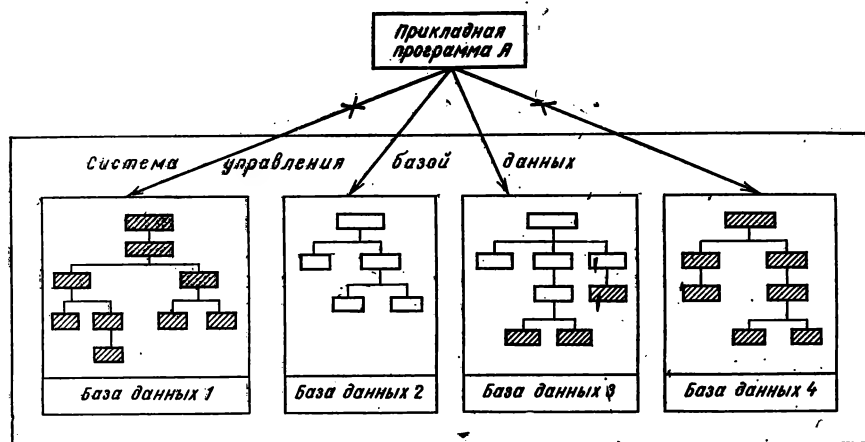
Независимость данных существенно облегчает процесс разработки прикладных программ. Упрощается также и их модификация. Программисту не требуется знание полной структуры данных в базе. Работа по модификации программы облегчается благодаря уменьшению сложности связей между программами в СОД. Даже если структура данных изменяется, объем работ по перепрограммированию существенно сокращается по сравнению с ранее существовавшими методами работы с данными, а во многих случаях перепрограммирование вообще не требуется. Появляется возможность расширять структуру данных и при этом перекладывать только программы, имеющие доступ к изменяемой части данных. И наоборот, можно удалять информацию из общей структуры данных, изменяя только программы, использующие эту информацию.

Другое преимущество применения банков данных состоит в сокращении, а в некоторых случаях — в исключении дублирования данных. В связи с этим удастся уменьшить дублирование тех частей программ, которые обеспечивают получение данных и занесение их в базу. С применением СУБД удастся также исключить большое число временных и промежуточных массивов информации, которые часто создавались при решении сложных задач.

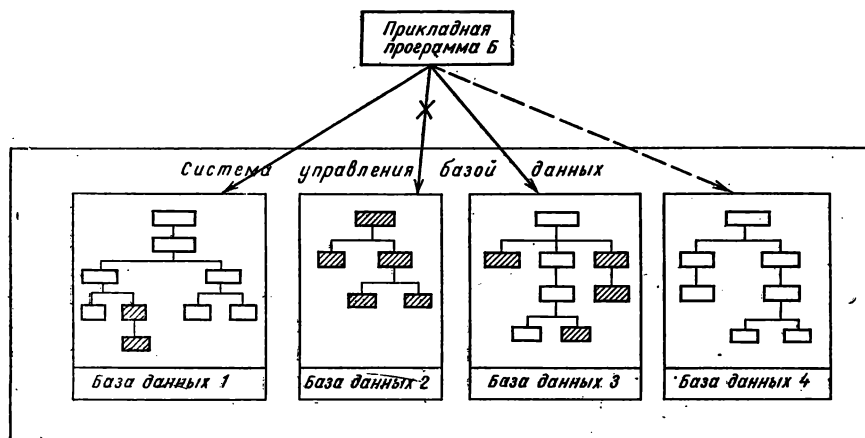
Преимуществом банков данных является также возможность защиты данных. Хотя система не может выявить ошибки недостоверности тех или иных элементов данных, некоторые СУБД обеспечивают целостность содержимого базы при попытке обновления одних и тех же элементов данных несколькими пользователями. В случае, если один пользователь извлекает информацию с целью ее обновления, система может закрыть доступ к этой информации для всех других пользователей, имеющих целью также модифицировать ее. Доступ закрывается до возвращения первым пользователем исправленных данных обратно в базу. Преимущества такого метода защиты данных проявляются, например, в случае использования банков данных при резервировании и продаже билетов на авиалинии. Если при обработке некоторого запроса не требуется модификация данных, некоторые системы разрешают доступ к информации даже во время ее обновления по другому запросу.

Обеспечивая защиту данных, СУБД не только управляет порядком обработки запросов, но и контролирует правильность самих запросов. Функции контроля реализуются в системе двумя способами.

При наличии в системе обработки данных нескольких баз данных необходимо оговорить, к каким из них программа имеет доступ. Кроме того, в пределах конкретной базы данных программа может не иметь доступа ко всем сегментам логической записи



а



б

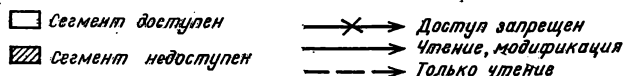


Рис. 20. Защита данных при использовании банка данных различными прикладными программами

(рис. 20). При этом ограничиваются ее возможности и в плане использования сегментов. Так, некоторые программы могут иметь доступ к определенным сегментам данных только с целью выборки информации, но не могут ее модифицировать. Описание баз данных для прикладной программы содержит всю необходимую инфор-

мацию о сегментах, доступных программе, и о характере доступности. Прежде чем предоставить программе требуемые данные, СУБД контролирует правильность запроса по описанию. Таким образом, имеется возможность защиты любого сегмента логической записи от непредусмотренного доступа прикладной программы.

Второй способ контроля правильности запросов обеспечивает защиту данных от доступа пользователей. С помощью специальных паролей можно контролировать доступ пользователей к тем или иным прикладным программам, имеющимся в системе обработки данных и работающим с базой данных.

При наличии в СОД банка данных упрощается разработка программ однократного применения и обеспечивающих программ вообще. Как правило, система пакетов прикладных программ, основанных на банках данных и включенных в СОД, не обеспечивает решения всех задач пользователя. В этом случае появляется необходимость в разработке собственных обеспечивающих программ. Такие программы с позиций их требований к организации данных ничем не отличаются от крупных базовых программ, реализованных в форме пакетов. С другой стороны, привязка новых программ к массивам, предназначенным специально для работы этих программных систем, может оказаться достаточно сложной в связи с необходимостью детального знания принципов организации таких массивов. В этом плане банки данных обеспечивают удобные средства для разработки программ, не зависящих от каких-либо других программ системы. Отметим также, что при необходимости оперативной разработки программ однократного применения пользователь может при работе с банком данных применять языки высокого уровня. При этом используются обычные трансляторы для этих языков и никакие модификации трансляторов не требуются.

Практически применение СУБД, обеспечивающей средства для построения и ведения базы данных, позволяет централизовать процесс управления данными в интегрированной системе и тем самым упорядочить потоки информации в ней. Это качество банков данных является чрезвычайно важным. Оно создает предпосылки для планомерного развития как каждой конкретной системы обработки данных, так и общей идеологии построения таких систем в целом.

2. 2. 6. Некоторые вопросы разработки и эксплуатации баз данных

Важнейшей задачей, определяющей эффективность работы универсальной системы управления базами данных и, следовательно, всей системы обработки данных в целом, является организация данных. Эта задача решается на этапе проектирования структуры базы данных.

Следует отметить, что при организации данных в базе необходимо свести до минимума дублирование данных. Для достижения общей эффективности системы это требование нужно сочетать

с другими требованиями, предъявляемыми к банку. Иногда предполагается накапливать данные в банке без дублирования. Однако при этом обычно не учитываются дополнительные затраты на проектирование. Безоговорочное соблюдение указанного требования может привести к существенному увеличению затрат при проектировании, определяемых, например, необходимостью установления дополнительных связей между отдельными массивами. Кроме того, усложняется обращение прикладных программ к банку данных, что может существенно увеличить машинное время, необходимое для обработки данных, и уменьшить надежность СОД в целом. С другой стороны, при неограниченном дублировании данных эффективность системы существенно падает и повышаются затраты на службу внесения изменений. Таким образом, следует искать компромиссное решение, которое, учитывая вышеназванные обстоятельства, смогло бы обеспечить минимум затрат при проектировании и эксплуатации банка данных.

Обновления баз данных осуществляются путем добавления новых записей и модификации записей уже имеющихся в массивах. Подобные обновления могут являться результатом работы прикладных программ или обработки запросов пользователя, сформулированных с применением языковых средств банка данных.

При необходимости изменения структуры существующих баз данных или организации дополнительных связей между массивами требуется реорганизация баз данных. Новая база данных описывается на языке системы с целью определения требуемых изменений. Дальнейшая реорганизация базы данных осуществляется автоматически с использованием программ системы управления базой данных. После проведения реорганизации база данных вновь готова к эксплуатации. Если структура записей, имевшихся в массивах до реорганизации, и их связи не были нарушены, ранее созданные программы могут работать с новой базой данных без каких-либо модификаций.

Обычно определение новой базы данных производится таким образом, чтобы изменения не затрагивали существующих прикладных программ. Однако, если это по ряду причин невозможно, модификации должны быть произведены лишь для тех программ, для которых произошли изменения в используемых данных. При этом модификации затрагивают лишь ту часть программ, которая представляет описание операций, необходимых для обеспечения программы исходными данными.

С точки зрения организации процесса эксплуатации интегрированной системы обработки данных обновление и реорганизация базы данных могут составить основу деятельности специально выделенного персонала*. При этом от правильного планирования структур баз данных, оперативного внесения обновлений в существенной мере зависит эффективность всей интегрированной системы.

* В зарубежной практике особо выделяется категория специалистов, называемая «администратор баз данных».

2. 3. СИСТЕМЫ ОБРАБОТКИ ДАННЫХ В НЕКОТОРЫХ ДРУГИХ СФЕРАХ ПРИМЕНЕНИЯ

Необходимость автоматизации процессов работы с данными существует не только в автоматизированных системах управления. Рассмотрим некоторые специальные сферы применения ЭВМ, их требования к организации данных и особенности предназначенных для этого программных систем.

2. 3. 1. Информационно-поисковые системы

Одной из широко распространенных сфер применения систем автоматизированной организации и поиска информации является библиотечное и архивное хозяйство [44], [45], [47]. Основная задача таких систем в этой сфере применения — поиск сведений о необходимых книгах и документах по заданному описанию, характеризующему требуемый материал с той или иной степенью полноты. База данных такой системы содержит массив материалов, записанных в форме аннотаций, отдельных ключевых слов или полного текста. Запросы на документы формулируются с применением специальных языковых средств и могут носить достаточно общий характер. Например, пользователя могут интересовать книги по истории культуры народов Японии VIII—XI веков. При этом перечень литературы по данному вопросу может быть достаточно большим, а причастность того или иного материала к данной тематике не всегда может быть определена с полной достоверностью. Система должна автоматически идентифицировать документы, удовлетворяющие поисковому запросу, и выдавать информацию о них пользователю. При этом может быть произведена выдача библиографических данных необходимого материала, либо его аннотаций (или реферата), либо дословного текста, если система хранит тексты документов и запрос требует его непосредственной выдачи. Подобные системы в отличие от банков данных часто называют информационно-поисковыми.

Информационно-поисковые системы и банки данных, обладая целым рядом общих свойств по структуре и принципам функционирования, имеют определенные различия, обусловленные спецификой решаемых задач. Банк данных для автоматизированных систем управления представляет средство централизации информации в системе и обеспечивает независимость прикладных программ от физической организации данных. При этом характеристиками его работы являются параметры, описывающие степень удовлетворения системы заданным требованиям (ограничение дублирования данных, возможности защиты данных и т. д.). Не все эти параметры легко могут быть выражены количественно, однако даже качественный анализ отдельных факторов, влияющих на эффективность, поможет в оценке той или иной конкретной системы.

СУБД, как программная система и часть системы обработки данных, в основном не предназначена непосредственно для

решения задач пользователя. Ее необходимость обусловлена спецификой эксплуатации современных вычислительных систем и определенными требованиями к самому процессу обработки данных, возникающими у пользователей. Информационно-поисковые системы, напротив, непосредственно предназначены для решения задач пользователей. В силу определенной замкнутости самой решаемой задачи подобная система может являться обособленной частью конкретной системы обработки данных или же при достаточно большом объеме и числе решаемых ею задач она может полностью монополизировать вычислительные средства и самостоятельно составлять систему обработки данных. Так, создание информационно-поисковой системы для книг и материалов Всесоюзной государственной библиотеки им. В. И. Ленина в Москве является самостоятельной задачей большой сложности.

Задачи, решаемые информационно-поисковыми системами, обуславливают и некоторые особенности программного обеспечения, предназначенного в первую очередь для осуществления поиска материалов по запросам. Функция поиска — идентификация документа по данному запросу — в сущности составляет важнейшую задачу таких систем, а неполная детерминированность этой задачи обуславливает сложность программной реализации. К настоящему моменту разработано много методов и средств, позволяющих строить подобные системы [44], [45], [47], однако их подробное описание не является предметом данной работы.

Говоря о создании информационно-поисковых систем, отметим, что в соответствии с требованиями, предъявляемыми к таким системам (в частности, в связи с требованием массового доступа к ним широкого пользователя), единственной структурой программы, обеспечивающей их реализацию, является программная система.

Вследствие перечисленных особенностей эффективность информационно-поисковых систем оценивается с помощью специальных критериев, важнейшими из которых являются полнота и достоверность выдаваемой по запросу информации [42]. Эти критерии также достаточно трудно поддаются количественному определению, однако их качественный смысл очевиден. Полнота ответа на запрос представляется отношением числа выданных по запросу релевантных (т.е. действительно удовлетворяющих запрос [42]) документов или информации о них к общему числу релевантных документов, которые содержатся в системе. Под достоверностью ответа на запрос понимается отношение числа выданных релевантных документов к общему числу документов, выданных по запросу. Количественные значения коэффициентов полноты и достоверности информационно-поисковых систем можно определять путем введения специальных методов усреднения коэффициентов, полученных при реализации конкретных запросов. Однако для нас в данном случае важно то, что для информационно-поисковых систем существует принципиальная возможность и даже необходимость введения этих характеристик, обусловленная требованиями решаемых задач.

Для оценки банков данных подобные критерии теряют смысл, поскольку полная детерминированность этих систем и требования, предъявляемые к ним с позиций сферы применения, допускают их использование в системе обработки данных лишь тогда, когда значения обеих характеристик равны 100% (чего не удастся добиться в сколько-нибудь сложной информационно-поисковой системе). В процессе работы банков данных появление при ответе системы на запрос не всей информации, требуемой запросом и имеющейся в системе, можно рассматривать как сбой в ее работе. Точно так же оценивается появление при ответе информации, не имеющей отношения к запросу. Таким образом, приведенные характеристики помогают выявить различия в двух типах систем организации и поиска данных: банках данных и информационно-поисковых системах.

2. 3. 2. Универсальные системы управления базой данных и передачей данных

Особенности внутренней реализации прикладной программы и ее использования могут быть обусловлены не только конкретным классом задач пользователей, но и определенными режимами эксплуатации вычислительных систем. При этом программы, предназначенные для работы в различных режимах, не всегда служат непосредственно для поддержания данного режима работы, т. е. не всегда их можно классифицировать как программы, расширяющие возможности основных операционных систем. Они могут иметь целью решение прикладных задач пользователей в определенных режимах применения ЭВМ. Подобные задачи возникают в процессе эксплуатации вычислительных систем, например в режиме разделения времени, когда несколько пользователей одновременно эксплуатируют вычислительные средства одной и той же вычислительной системы, пользуясь абонентскими пунктами (локальными или удаленными). При этом различают два случая использования ЭВМ в таком режиме.

В первом случае пользователи связаны друг с другом общими информационными массивами, обрабатываемыми в процессе решения задач. Примером такой системы может служить система связи отраслевой АСУ с автоматизированными системами отдельных предприятий и организаций для формирования плановых и отчетных документов по отрасли, контроля выполнения планов и т. п.

Здесь необходимость использования банков данных обусловлена всеми перечисленными выше требованиями взаимосвязи решаемых задач. Кроме того, при совместной эксплуатации одних и тех же баз данных различными пользователями существенно повышаются многие из этих требований, например требование защиты данных.

Для решения такого рода задач применяются универсальные системы, обеспечивающие как обслуживание баз данных, так и управление передачей данных на основе использования специаль-

ных телекоммуникационных методов доступа, входящих в состав операционных систем. В этом случае основу системы управления базой данных и передачей данных составляет управляющая программа. Эта программа обеспечивает связь ЭВМ с главным терминалом, через который осуществляется управление системой обработки данных, и со всей сетью присоединенных к машине удаленных терминалов, через которые достигается доступ к базам данных. Управляющая программа выполняет прием, передачу и обработку поступающих запросов, вызов и передачу управления соответствующим прикладным программам. Данная система с позиций архитектуры представляет собой программную систему, в которой управляющая программа приобретает большое самостоятельное значение. Таким образом, здесь универсальная система управления базами данных и передачей данных представляет неотъемлемую часть системы обработки данных.

Во втором случае отдельные пользователи обособлены и применяют собственные информационные массивы. На практике одна и та же вычислительная система может обслуживать одновременно пользователей обоих типов и мы выделяем эти группы лишь с целью анализа специальных требований каждой из них к системе обработки данных.

Необходимость системы организации данных в этом случае обусловлена требованием сокращения объемов данных, передаваемых по линиям связи. Это достигается путем централизации используемых программ и массивов информации в системе обработки данных. Обеспечивая централизацию информации, системы управления базами данных и передачей данных позволяют уменьшить число и объем программ, необходимых для обслуживания массивов данных, с которыми пользователи постоянно работают. Кроме того, подобные системы сокращают среднее время обработки запроса, которое включает и время передачи сообщений.

Отметим, что в обоих случаях важнейшими критериями оценки эффективности рассматриваемых систем являются надежность и среднее время обработки запроса. Первостепенная важность этих характеристик определяется основными требованиями к вычислительным системам, работающим в режиме разделения времени.

2. 3. 3. Запросно-ответные системы

Критерий времени ответа является особенно важным в тех случаях, когда вычислительная система работает в реальном масштабе времени. По самому определению таких систем время ответа на запрос здесь должно быть достаточно коротким. Примером такой системы может служить система распределения билетов на авиарейсы. Эти и подобные им системы представляют интерес, поскольку они связаны с определенным режимом эксплуатации ЭВМ и достаточно широким кругом задач пользователей. С другой стороны, конкретная специализация этих систем (запросно-ответ-

ные системы) проявилась в их структуре и принципах эксплуатации.

Запросно-ответные системы ориентированы на определенный класс решаемых задач. Это обусловлено требованием их максимальной эффективности при работе вычислительной системы в реальном масштабе времени. Поэтому подобные системы имеют обычно жесткую структуру баз данных, обусловленную требованиями традиционных задач данного класса. Системы управления базами данных для запросно-ответных систем характеризуются тем, что они помимо программ, предназначенных для обслуживания данных, содержат программы для их непосредственной обработки по прикладным задачам. Таким образом, запросно-ответные системы могут представлять собой самостоятельные системы обработки данных в смысле замкнутости подобных систем с позиций их применения. Языковые средства для работы с запросно-ответными системами обеспечивают не только возможности обслуживания данных, их поиска и выдачи по запросам, но и возможности обработки данных в соответствии с задачами конкретной системы.

Запросно-ответные системы обычно предназначаются для использования в режиме разделения времени с большим числом пользователей. Применение этих систем в режиме пакетной обработки хотя и допустимо, но неэффективно, так как утрачивается основное их преимущество перед другими системами организации данных — оперативность обработки информации и ответа на запрос. В то же время сама обработка данных в запросно-ответных системах в среднем проще, чем обработка данных в пакетном режиме.

Функции запросно-ответной системы управления базами данных принципиально могут быть реализованы универсальными системами управления базами данных после соответствующей их генерации и специальной привязки к определенному классу решаемых задач. Однако эти системы содержат большое число средств, обеспечивающих дополнительные по отношению к запросно-ответным системам функции. Такие средства зачастую не требуются в запросно-ответных системах и их наличие снижает эффективность эксплуатации этих систем. Универсальные системы управления базами данных не всегда обеспечивают основное требование к запросно-ответным системам — оперативный ответ на запрос.

Исходя из изложенного системы организации данных можно разделить по типам в соответствии со сферами применения, структурой, принципами функционирования и эксплуатации конкретных программных реализаций (табл. 1).

2. 4. ТРЕБОВАНИЯ СИСТЕМ ОРГАНИЗАЦИИ ДАННЫХ К ВЫЧИСЛИТЕЛЬНЫМ СРЕДСТВАМ

Рассмотрим специальные требования, предъявляемые к вычислительным системам с позиций реализации систем обработки данных, на примере банка данных для работы СОД в автоматизированных системах управления.

Таблица 1

НЕКОТОРЫЕ ТИПЫ СИСТЕМ ОРГАНИЗАЦИИ ИНФОРМАЦИИ И ИХ ХАРАКТЕРИСТИКИ

Типы систем организации информации	Эксплуатационные характеристики			
	возможности проектирования структур баз данных	возможность работы в режиме теле-обработки	место в СОД	основные критерии эффективности
Системы с жесткой структурой обслуживаемых баз данных	Жесткая структура	В зависимости от реализации	Связь с другими программами	Неизбыточность данных; дополнительные критерии в соответствии с требованиями решаемых задач; надежность
Универсальные системы управления базами данных	Проектируемая структура	В зависимости от реализации	Связь с другими программами	Неизбыточность данных; дополнительные критерии в соответствии с требованиями решаемых задач; надежность
Запросно-ответные системы	Жесткая структура (реже — проектируемая структура)	Стандартная	Замкнутая	Среднее время ответа; надежность
Информационно-поисковые системы	Жесткая структура (реже — проектируемая структура)	В зависимости от реализации	Замкнутая	Полнота; достоверность; надежность

Современные интегрированные системы обработки данных характеризуются большими объемами информационных массивов. Например, информационный массив кадрового состава на предприятии, насчитывающем несколько тысяч сотрудников, может иметь объем в сотни и тысячи килобайт*. Для учета внутренних ресурсов среднего предприятия (запасных частей, инструментов и др.) необходимо хранение многих мегабайт информации. Банк данных отраслевой АСУ, в котором должны найти отражение связи между десятками и сотнями предприятий, может достигать по объему десятков и сотен мегабайт.

Таким образом, информационные массивы в современных АСУ могут иметь объем от нескольких мегабайт до нескольких сотен мегабайт и более. При этом задачи оперативного управления и фоновые** задачи неизбежно сочетаются в рамках одной АСУ. Чтобы время ответа на запрос (или на определенную категорию запросов) не зависело существенно от неравномерности поступле-

* Байт — восьмизначная единица меры памяти в ЕС ЭВМ. 1К байт = 1024 байта; 1М байт = 1024К байта.

** Фоновые задачи — низкоприоритетные задачи; в режиме мультипрограммирования они выполняются в такие временные промежутки, когда работа высокоприоритетных программ по каким-либо причинам невозможна.

ния запросов, в частности от перекрытия времени их обработки, возникает необходимость резервирования ресурсов вычислительной системы, на которой реализован банк данных. При наличии же резервных ресурсов встает проблема их эффективного использования запросами с меньшим приоритетом, выполнение которых может быть отложено при поступлении высокоприоритетных запросов. Все эти условия определяют специальные требования к техническим средствам и средствам организации вычислительного процесса, на базе которых реализуются банки данных для АСУ.

Среди технических средств при разработке банков данных особую роль играют накопители информации. При выполнении запросов к банку данных вычисления обычно незначительны или вообще отсутствуют. Основными операциями являются поиск в массивах базы данных для подготовки ответа и обновления данных, пересортировки и перекомпоновки, связанные с реорганизацией баз данных. При этом поиск нередко ведется по многим параметрам. Состав этих параметров изменяется от одного поиска к другому в рамках одного запроса, а также между запросами. Если массивы информации размещены на магнитной ленте, то в процессе поиска неизбежны длительные перемотки ленты. Например, только последовательный просмотр семи мегабайт информации на магнитной ленте с использованием современных лентопротяжных механизмов потребует более 2 мин. Затраты времени на перемотку возрастут при использовании одной и той же ленты одновременно для подготовки ответов на несколько связанных между собой запросов. Поэтому практически невозможно обеспечить требования по времени ответа на запросы путем использования в банках данных для автоматизированных систем управления только накопителей на магнитной ленте. Проблемы хранения информации и доступа к ней в банках данных АСУ можно решить путем сочетания использования накопителей на магнитной ленте и накопителей прямого доступа — магнитных дисков и магнитных барабанов [2], [4].

До последнего времени в арсенале внешних запоминающих устройств для отечественных ЭВМ отсутствовали достаточно емкие и надежные накопители прямого доступа, что не позволяло осуществлять разработку мощного банка данных для нужд АСУ. В настоящее время в комплексе технических средств ЕС ЭВМ предусматривается наличие магнитных дисков (постоянных и сменных) в качестве стандартных средств любой ЭВМ Единой системы. Накопители со сменным пакетом магнитных дисков емкостью 7,25 Мбайт (ЕС-5050, ЕС-5052, ЕС-5056 и др.) обеспечивают доступ к любой записи максимум за 150 мс, причем среднее время доступа составляет 75 мс. Вся информация, записанная на пакете магнитных дисков, может быть последовательно просмотрена менее чем за минуту. Таким образом, это быстрее, чем аналогичная операция с магнитной лентой. Однако пакет магнитных дисков позволяет использовать специальные методы организации и доступа к данным, при которых поиск осуществляется без последователь-

ного просмотра всего массива информации*. При этом скорость поиска возрастает в сотни раз, что создает предпосылки для удовлетворения потребностей оперативного управления и использования вычислительных систем в реальном масштабе времени.

С другой стороны, следует иметь в виду, что хранение всех данных на носителях с прямым доступом экономически неоправдано, так как некоторые данные в АСУ являются архивными и, таким образом, редко используемыми. Они не требуют постоянной обработки и обновления, а потому могут храниться на более дешевых носителях информации, например на магнитных лентах. Такое разделение данных по частоте использования и их важности для обработки целесообразно. При этом улучшаются возможности экономии оперативной и внешней памяти, сокращается время выборки и повышается эффективность эксплуатации банка данных.

Кроме накопителей информации, технические средства АСУ должны включать высокоэффективные средства сбора данных, ввода запросов в систему и доставки ответов к месту их использования. Только при наличии этих средств можно обеспечить автоматизированное управление быстротекущими процессами и оперативное управление с участием человека. Благодаря различным терминальным устройствам, выполненным в виде пишущих машинок, алфавитно-цифровых и графических индикаторов и других средств, предусмотренных в ЕС ЭВМ, достигается удобный доступ к банку данных пользователей различной квалификации.

Операционные системы ЕС ЭВМ также создают широкие возможности для организации на их основе банков данных и автоматизированных систем управления. Поскольку любая система управления базой данных представляет собой по архитектуре программную систему, разработка управляющей части подобной системы, роль которой для этого класса программ особенно высока, упрощается в связи с возможностями использования средств управляющих программ, входящих в состав операционных систем ЕС ЭВМ. Следует специально отметить ту часть управляющих программ операционных систем, которая реализует организацию и доступ к данным, расположенным на внешних носителях информации, — управление данными. Средства управления данными обеспечивают основу для создания СУБД и представляются особенно эффективными для организации работы с информацией, расположенной на устройствах прямого доступа [17].

Кроме использования управляющей программы, система обработки данных АСУ, и в частности банк данных, может использовать такие программы, входящие в состав операционных систем ЕС ЭВМ, как высокоэффективную программу сортировки-объединения, утилиты для преобразования данных на внешних носителях (например, для переноса данных с пакета магнитных дисков на ленту и обратно) и др.

* В первую очередь следует отметить индексно-последовательный метод доступа, широко применяемый при разработке систем управления базами данных [2], [17], [19].

Глава 3

НЕКОТОРЫЕ ВОПРОСЫ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ

Разработка прикладной программы — комплекс научно-исследовательских и опытно-конструкторских работ. Эффективность эксплуатации вычислительных средств существенно зависит от качества системы прикладного программного обеспечения. При этом прикладные программы имеют тенденцию к усложнению как по структуре, так и по принципам функционирования. Это вызвано рядом причин, важнейшими из которых являются следующие:

увеличение объема и сложности задач, решаемых с помощью прикладных программ;

необходимость создания типовых программ, одинаково применимых различными пользователями на различных конфигурациях технических средств и операционных систем;

необходимость обеспечения простоты эксплуатации программы с целью привлечения к работе с вычислительной системой большого числа пользователей и расширения таким образом сфер применения.

Тенденция к расширению объема и усложнению прикладного программного обеспечения обуславливает тот факт, что затраты на создание прикладных программ существенно возрастают. В настоящее время разработка программного обеспечения сложной интегрированной системы обработки данных стоит в 1,5—2 раза дороже технических средств, необходимых для ее эффективного функционирования *. Практически в этой ситуации разработку прикладных программ следует рассматривать как сложный комплекс научно-исследовательских и опытно-конструкторских работ. Такой подход тем более важен при создании прикладного программного

* "Datamation", vol. 19, № 5, p. 48—59.

обеспечения для ЭВМ Единой системы, где программная совместимость отдельных вычислительных машин обеспечивает возможность массового применения комплексов прикладных программ.

В связи с этим в процессе разработки и применения прикладных программ возникает ряд задач, важнейшими из которых являются:

- организация управления разработкой пакетов прикладных программ;

- разработка технической и эксплуатационной документации в ходе создания программы;

- обеспечение высокой технико-экономической эффективности программных реализаций;

- внедрение и эксплуатация программ.

Рассмотрим эти вопросы на примере создания прикладных программ, имеющих архитектуру программной системы, поскольку при разработке этот вид программ представляет наибольшую сложность, а в эксплуатации он получает широкое распространение в связи с расширением сфер применения вычислительных машин.

Существуют определенные общепринятые этапы разработки технических систем. При этом каждый этап заканчивается созданием технической документации, в которой формулируются результаты работы. В процессе изложения мы будем придерживаться принятого деления этапов работ *.

3. 1. РАЗРАБОТКА ПРОГРАММ. ЭТАП ТЕХНИЧЕСКОГО ЗАДАНИЯ

Работы на этапе подготовки технического задания. На первом этапе создания программной системы проводится анализ класса задач, которые должны реализовываться с использованием программной системы, и существующих методов их решения. Выявляются основные требования к будущим программным реализациям, которые соотносятся с экономическими возможностями и людскими ресурсами организации, проводящей разработку. При этом следует учитывать ряд факторов, влияющих на эффективность результатов работы.

Связь разработчиков с пользователями. Для разработки гибкой программной системы необходимы специалисты, способные проанализировать задачи, которые должны решаться с помощью программы. Поскольку одни и те же задачи имеют определенную специфику решения для различных пользователей, специалисты-разработчики должны представлять себе направления возможных

* В настоящее время проводится комплекс работ по созданию системы Государственных стандартов, регламентирующих основные вопросы создания программного обеспечения ЕС ЭВМ — Единая система программной документации (ЕСПД). Используя существующие документы, определяющие порядок проведения разработок, мы основное внимание уделяем специфике проведения разработок прикладных программ,

модификаций системы в связи со спецификой решаемых задач. Если разрабатывающая организация специализируется на создании программных систем, желательно наличие в ней специалистов по различным сферам применения вычислительной техники и конкретным задачам, решаемым в этих сферах. В ином случае недостаточное знание рассматриваемых задач и их возможных модификаций должно компенсироваться привлечением к процессу анализа ряда пользовательских организаций, в системах обработки данных которых решаются задачи данного класса.

Иногда возможные методы решения задач и модификации этих методов либо уже сформулированы определенной группой пользователей, заинтересованных в разработке соответствующей программы, либо стандартизуются исходя из логики самой задачи и специфики областей, в которых она решается. Например, в нашей стране ряд научно-исследовательских организаций занят разработкой и внедрением сетевых методов планирования и управления, что служит предпосылкой для гибкой программной реализации этих задач. Другим примером может служить общая задача линейного программирования или ее модификации: транспортная задача, целочисленное программирование и др. В ряде книг и статей сформулированы основные методы решения этих задач, возможные их модификации в плане представления входных и выходных данных, а также введения каких-либо специальных методов обработки, например анализа точности полученного решения.

Общие формы и методы анализа прикладных задач с позиций их программной реализации существенно зависят от класса решаемых задач и в настоящее время еще не определены с достаточной четкостью. Однако некоторые рекомендации по вопросу анализа прикладных задач, например в АСУ, уже сформулированы [55], [58].

3. 1. 1. Выработка альтернативных решений

Результатом анализа решения задачи у различных пользователей является определение требований к программной реализации для обеспечения ее гибкости и некоторых других системных свойств. Однако односторонний учет повышенных требований может привести к неоправданному усложнению разрабатываемой программы, что повлечет увеличение времени и затрат на разработку, а в дальнейшем может отрицательно сказаться на эффективности эксплуатации программы. Поэтому требования к программной системе необходимо соотносить с важнейшими организационно-экономическими характеристиками разработки: сроками проведения разработки, общими затратами, имеющимися людскими ресурсами, квалификацией специалистов, связями с другими программными системами и др.

Опыт эксплуатации прикладных программ показывает, что увеличение затрат на разработку программы не всегда влечет за со-

бой повышение эффективности ее эксплуатации. Сложность самой разработки и определения эффективных форм эксплуатации прикладной программы не позволяют однозначно определить «оптимальные» соотношения затрат и эффективности. Поэтому результатом анализа, проводимого разработчиками системы, должен явиться набор альтернативных вариантов программной реализации. Каждый вариант реализации должен сопровождаться оценкой затрат и степени удовлетворения в некотором смысле идеальных (т. е. не зависящих от конкретных условий разработки) требований. В результате анализа желательно иметь более одного варианта решения, так как окончательное определение наилучшего варианта на этапе анализа вряд ли возможно, а выработка набора альтернатив облегчает процесс выбора решения на последующих этапах работы.

Основное содержание технического задания. Анализ программной реализации является важным этапом разработки системы, и его результаты во многом определяют эффективность всей последующей работы. Поэтому документы, описывающие процесс анализа и его результаты, должны составляться таким образом, чтобы к ним можно было легко обратиться на следующих этапах разработки, в особенности в тех случаях, когда они проводятся другими специалистами. В документах должны содержаться общие требования к реализуемой программной системе и вычислительным средствам, необходимым для ее работы. Такие общие требования, если они связаны с содержательной частью решаемой задачи и не зависят от того или иного альтернативного решения, делают разработку более определенной.

В техническом задании должны быть приведены альтернативные варианты конкретных решений вместе с предварительным определением затрат и эффективности этих вариантов. Описания вариантов решения задачи должны содержать сведения о характере входной и выходной информации, а также требования к обработке данных и предлагаемые методы их реализации.

Таким образом, основу документации на данном этапе составляют сведения, которые в дальнейшем послужат пользователю для оценки применимости программной системы при решении его задач. Однако в данном случае указанные сведения представляют собой общие рекомендации по реализации программы и их технико-экономическую оценку.

Кроме того, техническое задание может содержать предложения по организации проведения дальнейших этапов разработки, в частности, здесь могут быть перечислены стадии разработки прикладной программы, обусловленные спецификой решаемой задачи. Здесь же могут быть сделаны предварительные замечания по составу технических средств, необходимых для работы программы, и составу эксплуатационной документации, которая должна быть в дальнейшем разработана для пользователей.

3. 2. РАЗРАБОТКА ПРОГРАММЫ. ЭТАП ТЕХНИЧЕСКОГО ПРОЕКТА

Следующим этапом создания программной системы является техническое проектирование, непосредственно подводящее к этапу программирования. Рассмотрим основные задачи технического проектирования в их связи с предыдущей и последующей стадиями работ.

3. 2. 1. Разработка блок-схем алгоритмов

Техническое проектирование начинается с выбора одного из альтернативных вариантов решения задачи в качестве базового и определения алгоритмов для его реализации. При этом целесообразно, а в некоторых случаях и необходимо разрабатывать блок-схемы алгоритмов решения задачи поэтапно. На каждом последующем этапе части общего алгоритма, реализуемые отдельными блоками, детализируются и представляются в виде самостоятельных блок-схем (рис. 21). Обычно алгоритм можно представить в виде блок-схемы несколькими способами. Выбор того или иного варианта определяется разработчиком в соответствии с результа-

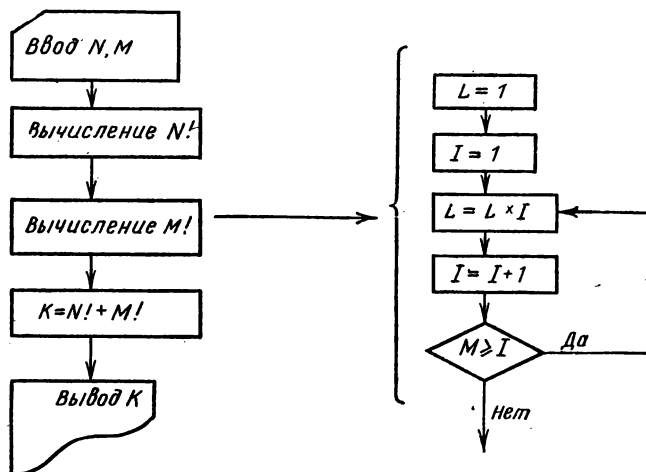


Рис. 21. Детализация блок-схемы алгоритма решения задачи

тами проведенного анализа, соглашениями между отдельными разработчиками, опытом этих специалистов и т. п.

На начальном этапе проектирования выделяются основные стадии обработки данных, а также методы управления обработкой, которые будут реализованы обособленными наборами модулей; устанавливаются требования к связям между этими модулями. В дальнейшем в процессе работы эти требования могут уточняться и корректироваться. В проработке более детальных блок-схем мо-

гут одновременно и независимо принимать участие несколько разработчиков. Это позволяет существенно сократить сроки проектирования путем привлечения к работе необходимого числа специалистов.

Таким образом, на любом этапе детализации блок-схемы алгоритма, кроме последнего, получается определенная разбивка всего процесса решения задачи на отдельные модули. Построение иерархии модулей представляет не побочный продукт для получения детального алгоритма, а имеет большое самостоятельное значение для всей программной реализации в целом. Хорошо спланированная модульная структура алгоритма решения задачи в дальнейшем находит отражение и в структуре самой программной системы, определяет ее гибкость и простоту модификаций. Кроме того, модульность системы, заложенная на этапе разработки блок-схемы, позволяет в дальнейшем привлечь к созданию программ достаточно большую группу программистов. При этом существует возможность регулировать разделение работы между ними в соответствии с уровнями их квалификации, поскольку в большой программной системе наряду со сложными высокоэффективными модулями имеется, как правило, много относительно простых программ.

В настоящее время разрабатываются* формальные методы расчленения схемы решения прикладной задачи на модули с целью получения наиболее эффективных программ, однако некоторые соображения можно высказать уже сейчас.

Более детальное расчленение алгоритма и программы на модули обеспечивает возможность отладки программы по частям до того, как она будет полностью разработана. Кроме того, в этом случае четкую форму приобретают логические взаимосвязи между частями программы, что в дальнейшем существенно облегчает работу программистов. С другой стороны, при слишком детальном расчленении программы необходимо уделять специальное внимание вопросам стыковки различных модулей в процессе разработки, в силу чего дальнейшую детализацию после определенного граничного уровня можно считать неэффективной.

Другим фактором, определяющим эффективность детального расчленения программы на модули, является число операций обмена между основной и внешней памятью ЭВМ, которое возрастает при увеличении числа модулей на определенном уровне. При этом увеличивается общее время решения задачи, что может существенно снизить эффективность программы.

В настоящее время еще нет достаточно отработанных формальных методов, определяющих группы шагов алгоритма, которые нужно объединять в общий модуль. Поэтому можно говорить лишь о том, что имеет смысл делить модули по функциональным признакам в соответствии с содержанием задачи или в соответствии с ее реализацией на вычислительной машине. В последнем

* См., например: Дал У., Дейкстра Э., Хоор К. Структурное программирование (Пер. с англ.). М., «Мир», 1975.

случае явно выделяются управляющие модули, модули ввода-вывода и модули других обслуживающих операций, а также модули обработки данных по алгоритму решения задачи. При этом модули обработки данных необходимо разрабатывать таким образом, чтобы части общего алгоритма, в наибольшей степени подверженные модификации, выделялись в отдельные элементы. Это позволит упростить и ускорить работу по модификации программной системы.

3. 2. 2. Организация исходных данных

С точки зрения использования данных разрабатываемая программная система может либо ориентироваться на некоторую систему управления базой данных, созданную ранее, либо содержать собственные средства хранения и поиска данных. Последнее наиболее эффективно в тех случаях, когда специфика задачи требует, например, при каждом применении программы использования в полном объеме всех обрабатываемых ею массивов данных, и сами данные должны поступать в определенной фиксированной последовательности. Однако даже в этом случае, когда функция поиска данных фактически отсутствует, необходимость объединения проектируемой программной системы с каким-либо банком данных может быть вызвана определенными требованиями, связанными с ее применением в интегрированной системе обработки данных. Важнейшим среди них является требование оперативного и точного обновления данных. Реализация этого требования существенно усложняется при дублировании информации.

Другой случай обособленного применения программных систем связан с задачами, которые не требуют ведения постоянных массивов данных, а работают каждый раз со вновь создаваемыми массивами. Примером такой системы может служить программа для решения задач линейного программирования. Вообще подобная специфика характеризует в большей мере методо-ориентированные программные системы нежели проблемно-ориентированные.

Вопрос об организации данных должен решаться именно на этапе технического проектирования. Решение этого вопроса существенно влияет на эффективность всей программной реализации. В процессе решения необходимо определить число исходных и промежуточных массивов данных, их организацию, внутренние связи и связи с другими массивами в базе данных, если программа ориентируется на применение банка данных и т. п. На этом этапе четко определяются структура и элементы входного языка, предназначенного для описания данных, форматы записей и элементы данных в отдельных массивах, а также общие алгоритмы преобразования данных из формы, принятой во входном языке, в форму, принятую в этих массивах.

В большинстве случаев программные системы работают с большими объемами исходных данных, что является одной из характеристик их сложности. При этом возникает задача контроля пра-

вильности исходной информации. Для облегчения такого контроля программная система может содержать специальные модули. Методы проверки правильности данных во многом зависят от принятых принципов построения языковых средств описания данных и обычно основываются на анализе соблюдения этих принципов. Один из наиболее распространенных способов контроля вводимой информации — редактирование — подробно описан в следующей главе в связи с программой, реализующей решение задач сетевого планирования и управления. Модули программной системы, предназначенные для контроля данных, должны не только выявлять имеющиеся при вводе данных ошибки, но и диагностировать их таким образом, чтобы пользователь мог легко внести исправления. Для этого специальные модули диагностики* должны установить характер ошибки в данных и выдать пользователю возможно более точное сообщение, содержащее описание ошибки и указание на запись, в которой эта ошибка встретилась. Здесь речь идет об ошибках в исходных данных, выявляемых до начала обработки. Поэтому общие диагностические средства, входящие в состав операционной системы [17], в данном случае недостаточны, поскольку ошибки в исходных данных могут быть выявлены операционной системой лишь на этапе обработки, да и то не во всех случаях. Однако, даже если операционная система обнаруживает ошибки, ее сообщения в процессе обработки неверных исходных данных не позволят быстро выявить и устранить ошибки вводимой информации. Таким образом, применение специальных диагностических средств является оправданным.

Если программная система строится на основе имеющегося банка данных, решение большинства вопросов, связанных с организацией данных, упрощается в силу того, что они регламентированы банком. Так, нет необходимости специально разрабатывать языковые средства описания данных, программы поиска данных и т. п. За разработчиком остаются лишь задачи формирования массивов в соответствии с требованиями банка, организации их связи с имеющейся базой данных (если она предполагается при разработке новой программы) и привязка программ обработки данных к системе управления базой данных. Все эти задачи решаются в соответствии с требованиями программной реализации и возможностями избранной системы управления базой данных.

Если программная система организует и ведет массивы данных самостоятельно, решение этих задач существенно усложняется. Преимущество такого варианта организации программы состоит в том, что разработчик, не связанный какими-либо внешними ограничениями, например возможностями того или иного банка данных, может разработать наиболее эффективную схему организации данных. При этом он может разработать новые методы доступа

* Заметим, что эти модули предназначены для диагностики ошибок в исходных данных при работе с программной системой и никак не связаны с тестовыми и диагностическими программами, которые упомянуты в первой главе.

и управления данными, не входящие в состав операционных систем. Однако следует иметь в виду, что такая программа, не удовлетворяющая стандартам, принятым в рамках программирования в ЕС ЭВМ, может оказаться слишком сложной для последующей модификации и потому недостаточно гибкой.

Окончательное решение об организации данных для работы программной системы разработчик должен принять исходя из сравнительной оценки эффективности и затрат при реализации того или иного варианта.

Анализ вопросов организации данных и модульной структуры программной системы показывает, что сложность и объем задач на этапе технического проектирования существенно больше, чем при простой разработке алгоритма решения прикладной задачи, которая обычно проводится при создании обеспечивающих программ. Практически на этом этапе закладывается идеология программной системы в целом и детально прорабатываются отдельные важные технические решения. Это необходимо представлять для правильной оценки соотношения этапов технического проектирования и программирования с позиций трудовых затрат и реализации цели создания программной системы вообще.

3. 2. 3. Выбор конфигурации технических средств

Любая прикладная программа предполагает для своей работы некоторый минимальный состав вычислительных средств в системе обработки данных пользователя. Сюда относятся минимальная конфигурация комплекса технических средств и конфигурация операционной системы. Для ЭВМ Единой системы минимальный состав технических средств определяется, как правило, минимальным необходимым объемом оперативной памяти, а также минимальным набором внешних устройств, включая устройства ввода-вывода и внешние запоминающие устройства, необходимые для работы программы. В некоторых случаях минимальный комплект должен включать какие-либо устройства, помимо входящих в стандартный комплект технических средств для традиционной пакетной обработки [4]. Например, при решении прикладной задачи может существовать необходимость применения графопостроителей, дисплеев и др. [4].

Минимальный комплект технических средств — понятие относительное. Так, при определенной организации данных и при использовании последовательного метода доступа накопителя на магнитных лентах и магнитных дисках могут оказаться взаимозаменяемыми устройствами [17], [19]. С другой стороны, ЕС ЭВМ обеспечивает пользователя спектром технических устройств каждого типа, отличающихся своими техническими и стоимостными характеристиками, но полностью технически и программно совместимых. В этом случае применение любого из них возможно в минимальном комплекте. Взаимозаменяемость устройств в минимальном комплекте должна специально оговариваться в эксплуатационной доку-

ментации на программную систему с тем, чтобы обеспечить наиболее широкий круг пользователей данной программы. Минимальная конфигурация технических средств, четко определенная и закрепленная в документации, позволяет выделить тот круг пользователей, вычислительные системы которых удовлетворяют требованиям данной программы.

Однако, обеспечивая принципиальную возможность решения задачи с помощью прикладной программы, минимальная конфигурация технических средств не гарантирует эффективность ее эксплуатации. Так, увеличение объема внешней памяти прямого доступа по сравнению с минимальной конфигурацией может обеспечить существенное повышение скорости решения задачи и сократить объем работы оператора по организации процесса выполнения программы. С другой стороны, минимальная конфигурация технических средств обычно обеспечивает решение задач лишь с ограниченным объемом исходных данных. Например, увеличение размерности матрицы при решении задачи расчета межотраслевого баланса на ЭВМ может вызвать необходимость увеличения объема оперативной памяти ЭВМ. В подобных случаях в эксплуатационной документации должны содержаться сведения о минимальной конфигурации вычислительной машины для решения задачи с различными объемами исходных данных.

Наряду с минимальной могут быть определены сведения об эффективной конфигурации для решения данной задачи. Однако, скорее, речь может идти не о наиболее эффективной конфигурации для выполнения программы, а о возможности эффективного использования дополнительных (по отношению к минимальному составу) технических средств, входящих в вычислительную систему пользователя. Рекомендации по данному вопросу должны быть включены в эксплуатационную документацию и использованы при установке и внедрении программы.

Определение минимальной конфигурации вычислительной системы, необходимой для выполнения программы, является задачей разработчиков на этапе технического проектирования. При этом основную роль здесь играют два связанных между собой критерия: эффективность программы, определяемая временем ее выполнения при некоторых нормативных объемах исходных данных, и требование гибкости программы в плане ее привязки к той или иной конфигурации технических средств. Одно из решений этой задачи состоит в обеспечении такой гибкости программной реализации, при которой минимальная конфигурация, необходимая для системы, невелика, но любое расширение комплекса технических средств может быть использовано для повышения эффективности программы без существенной ее модификации. Так, например, гибкость использования оперативной памяти ЭВМ может быть достигнута путем создания программных модулей системы со структурой запланированного перекрытия [17], когда программный модуль разделяется на отдельные части — сегменты. Один из сегментов — основной (корневой) — постоянно находится в оператив-

ной памяти и по мере необходимости вызывает в процессе работы остальные сегменты *. Вводимые сегменты (кроме корневого) занимают одну и ту же область оперативной памяти. Так, в примере, изображенном на рис. 22, объем памяти, занимаемый программой, определяется как сумма объема памяти, занимаемого корневым

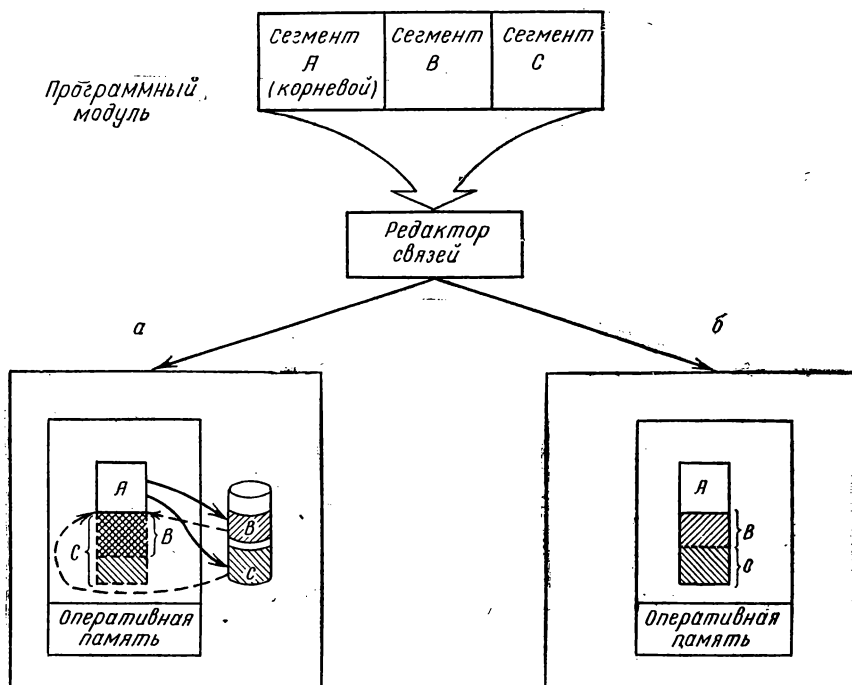


Рис. 22. Возможности использования оперативной памяти:
а — структура запланированного перекрытия; б — простая структура

сегментом, и объема, необходимого для максимального по размеру из загружаемых сегментов.

Структура запланированного перекрытия (или, иначе, структура с перекрытием) проектируется в процессе создания программного модуля. Окончательное формирование выполняемой программы производится редактором связей — специальной обрабатывающей программой, входящей в состав операционных систем ЕС ЭВМ [17], [19]. При этом редактор связей может быть использован как для создания прямой (обычной) структуры программного модуля, так и для создания структуры с запланированным перекрытием. Таким образом, используя редактирование (обработку программного модуля редактором связей), пользователь может получить

* Следует различать понятия сегмента и корневого сегмента для прикладных программ и для баз данных.

модуль, загружаемый в область оперативной памяти определенного объема.

Отметим, что в случае уменьшения состава технических средств (в данном случае при сокращении объема оперативной памяти) снижается эффективность работы программы. Так, в приведенном примере увеличивается число операций ввода программных сегментов и, следовательно, возрастает общее время решения задачи. Однако гибкость программной системы существенно расширяет возможности ее применения.

При создании подобных гибких программ к разработчикам предъявляются дополнительные требования. Различные варианты разработки должны рассматриваться с учетом затрат, необходимых для их реализации.

3. 2. 4. Определение конфигурации операционной системы

Определение конфигурации операционной системы, которая требуется для прикладной программы, имеет свою специфику. Стандарты программирования в ЕС ЭВМ [17] построены таким образом, чтобы обеспечить совместимость различных конфигураций операционной системы. Поэтому первое и наиболее существенное решение, которое должно быть принято в плане обеспечения определенного круга пользователей, — выбор типа операционной системы — ОС ЕС или ДОС ЕС. Такой выбор может быть осуществлен на этапе разработки технического задания.

Опыт создания прикладных программ зарубежными фирмами показывает, что большинство распространенных прикладных задач имеют для своего решения программы, предназначенные для обеих операционных систем. При этом возможности программ, например, в операционной системе OS/360 фирмы IBM обычно несколько выше, чем в DOS/360, но принципы построения программ и методы решения задач зачастую одинаковые. Так, программы для решения задач линейного программирования: Linear Programming System/360 (LPS/360) для DOS/360 и Mathematical Programming System/360 (MPS/360) для OS/360 близки по своим эксплуатационным возможностям и средствам их реализации. Различие проявляется в некоторых технических характеристиках программ. Так, максимальная размерность задачи для MPS/360 выше, чем для LPS/360. Это обусловлено возможностями операционных систем и конфигурациями технических средств, которые они могут эффективно обслуживать.

Фактически в рамках выбранного типа операционной системы речь идет в первую очередь о выборе тех или иных средств, предоставляемых ею. Эти средства (или возможности) можно разделить на возможности управляющей и обрабатывающих программ операционной системы.

Наиболее важные возможности управляющей программы: обеспечение различных режимов мультипрограммирования (с фиксированным или переменным числом задач) и в связи с этим — методов

распределения ресурсов вычислительной системы; различные методы организации данных на внешних машинных носителях информации и методы обмена данными между оперативной памятью и внешними запоминающими устройствами. Правильное использование этих возможностей определяет гибкость и эффективность всей программной реализации в целом.

Одним из важнейших вопросов при определении необходимых средств операционной системы является выбор языков программирования с учетом особенностей самих языков и имеющихся трансляторов. Общие тенденции использования языков высокого уровня при разработке сложных программных систем были рассмотрены выше. Выбор языков для конкретной разработки диктуется определенными экономическими соображениями. Эффективность программирования на языке Ассемблера потенциально в несколько раз выше, чем программирование на проблемно-ориентированных языках высокого уровня. Это связано с тем, что на языке Ассемблера разработчик получает доступ практически ко всем средствам, предоставляемым операционными системами и вычислительной машиной [17], [19], [23], в то время как для языков высокого уровня этот доступ в большей или меньшей степени ограничен (табл. 2) *.

Однако эффективность программирования на языке Ассемблера проявляется лишь в том случае, когда разработчик хорошо знает предоставляемые ему возможности, т. е. детально представляет себе принципы работы технических средств и операционных систем. Для получения действительно высоких результатов при программировании на машинно-ориентированном языке требуется более высокая квалификация программистов и разработчиков. В ином случае использование Ассемблера вряд ли целесообразно, поскольку трудоемкость разработки программ с его применением при прочих равных условиях выше, чем с применением проблемно-ориентированных языков программирования.

Таким образом, можно сформулировать некоторые соображения по вопросам применения языков программирования при проведении разработки программной системы. Вопрос использования того или иного языка программирования должен решаться отдельно для каждого конкретного модуля программной системы на различных уровнях детализации. При этом должны учитываться общие требования к обработке, проводимой данным модулем, и в том числе требования к его эффективности, которые определяются частотой выполнения модуля в процессе работы программной системы. Чем чаще будет выполняться модуль, тем эффективнее его разработка на языке Ассемблера (при определенной квалификации разработчиков). Отсюда, в частности, следует эффективность разработки управляющих программ системы на языке Ассемблера.

Если разработка должна быть проведена в сжатые сроки, а также в случаях, когда модули программной системы невелики по

*. Отметим, что наиболее широкие возможности по средствам разработки программ после Ассемблера имеет язык ПЛ/1,

**СОПОСТАВЛЕНИЕ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ
ПО НЕКОТОРЫМ ВОЗМОЖНОСТЯМ РАЗРАБОТКИ ПРОГРАММ**

Возможности при разработке программ	Языки программирования						
	Алгол	Кобол (F)*	Фортран		ПЛ/1 (F)*	ППГ	Ассем-блер
			G*	H*			
Организация программ простой структуры	Да	Да	Да	Да	Да	Да	Да
Организация программ со структурой запланированного перекрытия	Нет	Да	Да	Да	Да	Нет	Да
Обработка последовательных наборов данных	Да	Да	Да	Да	Да	Да	Да
Обработка индексно-последовательных наборов данных	Нет	Да	Нет	Нет	Да	Да	Да
Распределение оперативной памяти	Нет	Нет	Нет	Нет	Да	Нет	Да

* Буквы характеризуют конкретный транслятор с данного языка, имеющийся в составе ОС ЕС, по объему оперативной памяти, требуемой для его работы [17].

объему и не предназначены для частого выполнения, возможно применение проблемно-ориентированных языков. При этом выбор конкретного языка высокого уровня зависит в основном от характера решаемой задачи и требований к программной реализации.

Помимо трансляторов с языков программирования разработчик может воспользоваться другими обрабатывающими программами операционных систем, а также ранее разработанными пакетами прикладных программ. Например, он может использовать ту или иную систему управления базами данных с целью организации данных для работы программной системы. Использование ранее разработанных программ характерно при создании пакетов, предназначенных для работы в интегрированных системах обработки данных, где зачастую требуется взаимосвязь отдельных программ.

При разработке программных систем для АСУ широко применяется обрабатывающая программа сортировки-объединения, входящая в состав обеих операционных систем ЕС ЭВМ, а также программы-утилиты для ведения и обслуживания наборов данных [17], [19].

В ходе разработки технического проекта программной системы разработчик должен определить обрабатывающие программы, которые могут быть применены для решения данной задачи, и выбрать те из них, которые наилучшим образом соответствуют требованиям данной прикладной задачи. При этом необходимо учитывать влияние выбранных программных средств на структуру и принципы функционирования разрабатываемой программы. В некоторых случаях при большой гибкости используемых системных программ (например, программы сортировки-объединения) такое влияние может оказаться минимальным. Однако при использовании систем управления базами данных оно может быть сущест-

венным и требовать специальной привязки создаваемой программы к возможностям имеющихся программных средств.

Создание новой программной системы, ориентированной на совместную работу с пакетами, разработанными ранее, имеет и негативные стороны. Они состоят в том, что пользователь будет вынужден включать эти пакеты в свою систему обработки данных, даже если дополнительная программа ему специально не требуется или подобная программа уже имеется в его системе. В последнем случае пользователь, только приступающий к организации собственной системы обработки данных, оказывается в более выгодном положении. Решение этой проблемы может обеспечить система специально разработанных стандартов, регламентирующих организацию связей между отдельными программами. Разработка новых программных систем существенно упростилась бы при взаимозаменяемости программ, реализующих на ЭВМ решение сходных задач. Примером и базой для подобной стандартизации может служить система стандартных соглашений о связях между программными модулями, принятая в рамках программирования для ЕС ЭВМ [17].

3. 2. 5. Планирование модификаций

Не все варианты реализаций программной системы, указанные на этапе разработки технического задания, могут найти отражение в основной версии программной системы. Поэтому некоторые требования могут оставаться нереализованными, что сужает круг потенциальных пользователей программы. С учетом этого на этапе технического проектирования необходимо уделить особое внимание возможным побочным вариантам реализации программной системы. Для этого полезно выделить те модули программы, которые могут в дальнейшем часто подвергаться модификации, и определить направление их реализации. Среди возможных вариантов реализации таких модулей можно выделить следующие.

Во-первых, параметрическая настройка программных модулей. В этом случае программный модуль, реализующий этап обработки данных, наиболее подверженный изменениям, «избирает» тот или иной вариант обработки в зависимости от параметров, задаваемых пользователями. Такие параметры являются элементами языковых средств программной системы и общие правила их описания должны быть определены как средства запроса обработки практически в любой программной системе. Подобным образом могут быть реализованы важнейшие функции, определенные требованиями к данной программной системе.

Во-вторых, создание модулей, включаемых в программную систему и реализующих различные виды обработки в соответствии с одной и той же функцией обработки. Такой принцип построения программной системы носит название функциональной избыточности [16]. Данный метод широко используется в операционных системах. Например, в состав операционной системы ОС ЕС входит

несколько трансляторов с одного языка высокого уровня. При этом каждый транслятор имеет свои особенности при реализации функции перевода программ с исходного языка.

Дублируемые модули обычно определяют пункты задания и могут вызываться по своим именам, находясь в одной общей библиотеке. При использовании такого метода для создания гибких программ, естественно, требуются дополнительные затраты на разработку и программирование системы, однако он в ряде случаев более эффективен, чем предыдущий, поскольку не требует машинного времени на определение конкретного варианта реализации обработки данных по заданным параметрам. Обычно этот метод эффективен в тех случаях, когда изменения больших блоков программы требуются достаточно часто.

В-третьих, обеспечение программной системы средствами генерации. При этом, как правило, разрабатываются специальные языковые средства генерации, с помощью которых программист при установке системы может описать вариант программной системы, необходимый для конкретного применения. В процессе генерации программы специфика задачи может быть учтена настолько полно, насколько это обеспечивается при разработке программной системы. Обычно трудоемкость разработки генерируемых программных систем гораздо выше и требует привлечения высококвалифицированных специалистов. Поэтому подобный способ обеспечения гибкости программы применяется в тех случаях, когда разрабатываемая система планируется в качестве универсальной для решения широкого класса задач и обеспечения широкого спектра конфигураций технических средств (например, при проектировании универсальной системы управления базами данных).

Любой из приведенных методов может быть применен в конкретной разработке самостоятельно или в совокупности с другими методами. Если программная система не учитывает специфику решения задач отдельных пользователей, возможна модификация модулей силами разработчиков, специалистов по сопровождению или квалифицированных программистов пользовательских организаций. При этом в соответствии с общими соглашениями о стандартах программирования, соглашениями, принятыми при разработке данной программы, а также в соответствии с алгоритмом решения, могут быть добавлены новые и изменены имеющиеся модули программной системы.

3. 2. 6. Разработка методов отладки

Модульность программной системы позволяет вести отладку ее элементов обособленно, традиционными методами. Однако вся система в целом достаточно сложна и не представляется в виде простой суммы отдельных компонентов. Поэтому наряду с отладкой модулей в ходе разработки программы стоит задача тщательной отладки системы как единого целого. Сложность отладки систе-

мы заключается в том, что общее число различных вариантов работы программы может быть очень велико. Зачастую в процессе тестирования невозможно проверить все варианты. Так, в примере, показанном на рис. 23*, общее число вариантов работы программы может превысить 10^{10} . В этом случае на этапе технического проектирования системы необходимо определить варианты работы программы, которые в дальнейшем будут наиболее часто реализовываться в процессе ее эксплуатации и поэтому должны быть проверены в первую очередь. Эти варианты определяются из содержания самой решаемой задачи и принятого направления ее реализации. В соответствии с перечнем вариантов работы составляется набор тестовых примеров, отражающих будущие условия примене-

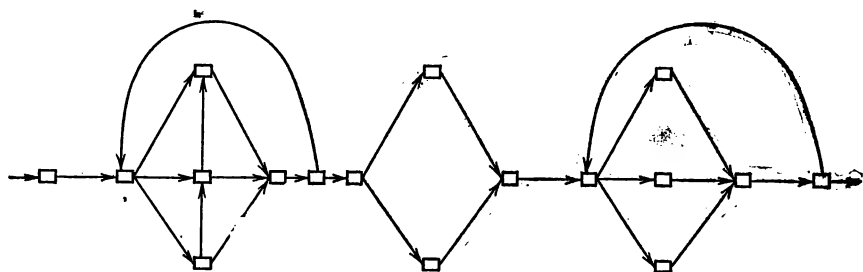


Рис. 23. Пример структуры программы. Дугообразные стрелки определяют циклы с количеством повторений, меньшим или равным 12.

ния. В частности, удобно использовать в качестве отдельных тестовых примеров те задачи, которые решаются или ранее решались у пользователей, имеющих опыт в реализации задач данного класса.

В дальнейшем, после программирования системы и выявления определенных ошибок в разработке (на этапе отладки), разработчики совместно с программистами устраняют их, последовательными итерациями приближаясь к окончательному варианту программной системы. При этом после каждой стадии исправления ошибок перечень тестовых примеров может корректироваться, если проведенные изменения каким-либо образом влияют на общую оценку различных вариантов работы программной системы.

Успешная отработка тестовых примеров еще не является полной гарантией высокого качества работы сложной программной системы, так как в ограниченный промежуток времени может оказаться невозможным проверить все варианты ее работы. В этом случае программа после разработки и отладки перед массовым внедрением должна пройти стадию опытной эксплуатации в системе обработки данных, например у нескольких квалифицированных пользователей, где подобные задачи решаются достаточно часто. В процессе опытной эксплуатации между пользователями и разработчиками должен поддерживаться постоянный контакт, в резуль-

* Рисунок взят из «Datamation», vol. 19, № 5, 1973, p. 48—59.

тате которого все замеченные ошибки в работе системы могут отслеживаться и исправляться в том же порядке, как это делалось на этапе отладки.

Отечественный и зарубежный опыт показывает, что даже в процессе массовой эксплуатации программные системы могут дорабатываться и выдерживать большое число версий с целью достижения наибольшей эффективности. Подобная систематическая работа по сопровождению сложных и дорогостоящих пакетов программ оправдана лишь в случае, когда в развитии вычислительных средств существует определенная преемственность, обеспечивающая длительный срок службы разработанной системы. Именно это свойство является определяющим для вычислительных машин Единой системы.

3. 3. РАЗРАБОТКА ПРОГРАММ. ПРОГРАММИРОВАНИЕ И ДОКУМЕНТИРОВАНИЕ

Программирование модулей системы. Программирование по заданным блок-схемам и алгоритмам представляет традиционный этап в создании прикладных программ. Однако этот процесс при разработке программных систем для ЕС ЭВМ имеет ряд особенностей. Сложность программных систем обычно такова, что на этапе технического проектирования неизбежно возникают ошибки в проработке отдельных модулей и частей программ. В связи с этим задачей этапа программирования наряду с реализацией системы является логический контроль правильности ранее принятых решений. Контроль осуществляется программистом как на стадии кодирования программ, так и в процессе их отладки. При этом некоторые изменения в ходе решения программист может вносить самостоятельно, в более сложных случаях требуется вмешательство разработчика.

В процессе программирования модулей системы программист определяет те места в программе, где логические ошибки в исходных данных могут вызвать ошибки в их обработке. В подобных случаях необходимо разработать дополнительные средства для диагностики логических ошибок, которые должны обеспечить локализацию ошибки и выдачу короткого и ясного сообщения, позволяющего пользователю быстро установить и устранить ошибку.

Этап программирования системы является развитием и продолжением этапа технического проектирования. В ходе программирования уточняется и дополняется идеология программной системы, заложенная ранее.

Программирование отдельных модулей ведется на языках, определенных на этапе разработки технического проекта. Для получения эффективных программных модулей программист должен знать не только соответствующий язык, но и особенности трансляторов с этого языка, входящих в состав выбранной операционной системы. В зависимости от этих особенностей различные варианты программной реализации алгоритма определяют различное качество работы модуля после его трансляции. Выбор того или иного

варианта программирования для каждого модуля зависит и от того, будет ли модуль каждый раз транслироваться перед выполнением или же после завершения разработки, трансляции и редактирования всегда будет использоваться полученный исполнимый (загрузочный) модуль [17], [19]. В последнем случае, который в практике разработки прикладных программ встречается чаще, программист может на разных стадиях разработки программы использовать различные трансляторы с данного языка. Как правило, операционная система содержит более одного транслятора для каждого языка программирования. При этом различные трансляторы совместимы по воспринимаемым программам, но обладают различными свойствами, обуславливающими эффективность их применения в определенных ситуациях. Так, транслятор Фортран G в ОС ЕС имеет ряд средств для отладки программ и диагностики ошибок программирования, а транслятор Фортран H обеспечивает высокую эффективность получаемых машинных программ [17]. В процессе разработки и отладки программы удобнее использовать первый транслятор, а после получения отлаженного модуля провести окончательную трансляцию с помощью второго транслятора.

В ходе выполнения программ системы на ЭВМ в соответствии с различными исходными данными пользователей и различными запросами на обработку может генерироваться большое число рабочих программ, отличающихся составом и последовательностью образующих их модулей. При этом в зависимости от запроса могут реализовываться различные варианты объединения модулей в рабочей программе. Здесь возникает проблема стыковки модулей, например, для передачи данных от предшествующего модуля к последующему. Поскольку число различных модулей, которые должны стыковаться в процессе решения задачи, может быть велико, на этапе программирования целесообразно выработать соглашения по организации связи между любыми модулями в рамках определенных совокупностей модулей системы. Следование подобным стандартам при разработке всех модулей программной системы обеспечит возможность независимой работы программистов при создании системы и позволит уменьшить число ошибок, связанных с несогласованностью отдельных частей программы. Выработка такого рода соглашений является одной из важнейших задач на этапе программирования.

Документирование программной системы. При создании программной системы особое внимание следует уделить разработке эксплуатационной документации, которая осуществляется на этапе программирования. Структура документации и ее содержание могут быть различным образом регламентированы в специальных стандартах. В связи с этим рассмотрим лишь основные сведения, которые должны быть в ней отражены. Предлагаемая ниже структура документации не является единственно возможной. Однако она основана на анализе имеющегося опыта документирования программных систем.

Вся эксплуатационная документация к программной системе может быть разбита на отдельные материалы в соответствии с двумя критериями:

уровень детализации сведений о программной системе должен соответствовать цели ее изучения;

каждый материал должен быть предназначен для определенного круга специалистов, связанных по характеру своей работы с программой.

Исходя из этого удобно разделить всю документацию по программе на «Описание применения», «Руководство оператора», «Руководство программиста» и «Руководство системного программиста».

«Описание применения» программной системы необходимо при оценке применимости системы для решения задач пользователей. Остальные документы определяют основные процедуры при работе с программой и необходимы в процессе ее использования. Поэтому документ «Описание применения» может тиражироваться и распространяться отдельно от остальных документов. В данном документе должны быть отражены следующие основные вопросы:

цели и основные задачи, для решения которых предназначена данная программа, сфера применения программы;

минимальная конфигурация вычислительной системы, требуемая для работы программы в различных вариантах и с разными объемами исходных данных; данные о типе и числе необходимых устройств ввода-вывода, необходимом объеме оперативной и внешней памяти, а также типе операционной системы, на применение которой ориентирована данная программа;

ранее разработанные программные средства или обрабатываемые программы операционной системы, требуемые для работы данной программы;

данные об эффективности работы программы в виде таблиц производительности на определенных нормативных объемах исходной информации и при определенных вариантах работы программы. Желательно, чтобы такие данные были приведены по результатам работы программы на различных моделях ЭВМ Единой системы и различных конфигурациях технических средств.

Документ «Описание применения» предназначен для специалистов, которые должны использовать программную систему в своей работе, а также специалистов, обслуживающих вычислительные системы пользователей. Например, если программная система предназначена для решения задач управления, первую группу специалистов составляют руководители предприятий, занятые решением данной задачи, и специалисты по управлению.

В документе «Описание применения» кроме подробного изложения задачи, для решения которой предназначена программная система, могут быть также приведены методы ее решения. Здесь целесообразно рассмотреть модели, положенные в основу решения задачи, а также границы их применимости. Изложение удобнее вести в терминах входной и выходной информации. Сведения по

обработке данных обычно даются лишь в связи с используемой моделью решения задачи, поскольку специалисты, на которых рассчитано данное руководство, могут не иметь необходимых знаний о вычислительной машине.

Характер информации, требуемой для работы программы, а также описание результатов обработки удобнее всего проиллюстрировать на демонстрационном примере, распечатки которого могут быть приведены в данном руководстве. Особенности программной реализации и принципы работы системы здесь приводить не следует. Если программа применяется в автоматизированных системах управления, то в этот документ могут быть включены рекомендации по организации решения данной задачи управления на предприятии в том плане, как это было определено на предыдущих этапах проектирования программы.

В документах «Руководство программиста» и «Руководство оператора» должны содержаться сведения о принципах работы программы и подробные инструкции по ее эксплуатации. Эти сведения не всегда могут быть четко разделены на сведения для оператора и сведения для программиста, поэтому мы приводим содержание данных документов без такого разделения. Отметим лишь, что в «Руководстве программиста» должна содержаться полная информация о всех возможных вариантах работы программы. Это необходимо для эффективной эксплуатации разработанных программных средств. «Руководство оператора» должно включать подробные инструкции по эксплуатации программы и взаимодействию с ней обслуживающего персонала. Важнейшие вопросы, излагаемые в этих руководствах, следующие:

основные логически обособленные этапы обработки данных. Обычно такие этапы реализуются в форме шагов одного задания [17], [19], состоящего в выполнении программной системы. Одновременно могут быть проиллюстрированы возможные потоки информации в программе как данные, переходящие от одного шага задания к другому. Кроме того, здесь четко прослеживаются функции модулей, реализующих выполнение каждого шага задания;

детальное изложение языковых средств, служащих для описания данных и запросов к обработке. Здесь описываются различные типы исходных данных, которые могут быть введены и обработаны программой, а также правила их описания с использованием имеющихся языковых средств. Необходимо также перечислить различные виды обработки данных, осуществляемые программой, и правила их запроса средствами языка запроса обработки;

описание исходных, промежуточных и основных массивов, обрабатываемых и составляемых программой. Особое внимание должно уделяться вопросам организации и обновления основных массивов данных. При этом, даже если данные обслуживаются другой программной системой, в руководстве должны содержаться основные сведения о процедурах работы с данными;

описание заданий для выполнения программной системы на

языке управления заданиями. Здесь должны быть указаны все управляющие инструкции, используемые при работе программы, а также случаи, в которых требуется их применение. Необходимо детально описать те управляющие инструкции, которые могут быть изменены пользователем в процессе работы (например, параметры из поля PARM оператора EXEC, передаваемые в программу и определяющие тот или иной вариант ее работы [17]);

все диагностические сообщения, выдаваемые системой, должны быть подробно разъяснены. Кроме того, необходимо определить действия обслуживающего персонала по устранению той или иной ошибки;

большинство сведений, содержащихся в этих руководствах, желательно проиллюстрировать одним или несколькими демонстрационными примерами, отражающими основные варианты применения программы. Исходные данные для демонстрационных примеров могут быть записаны на носителе информации вместе с модулями программной системы с целью облегчения их выполнения. Все вопросы, связанные с подготовкой и выполнением примеров, должны быть подробно рассмотрены в документации;

необходимо привести инструкции для вычисления объемов оперативной и внешней памяти, требуемых для каждого массива данных и для работы программы. Здесь же указываются сведения об организации массивов данных и взаимозаменяемости внешних носителей.

В документе «Руководство системного программиста» приводятся сведения о действиях, которые необходимо провести для привязки системы к конкретной задаче пользователя и конфигурации его вычислительной системы. Если в процессе привязки предполагается генерация программной системы, в данном руководстве должны быть приведены подробные сведения о ее проведении. Обычно система предоставляет широкие возможности реализации различных вариантов привязки. В этом случае в данное руководство необходимо включать рекомендации, обеспечивающие эффективную работу программы в условиях конкретного применения.

Эксплуатационная документация является неотъемлемой частью программной системы и содержит все сведения, необходимые для установки и использования программы в системе обработки данных. Однако более детальную информацию, обеспечивающую сведения, необходимые для модификации модулей программной системы, не обязательно включать в комплекс документации. Это связано с тем, что сложность программной системы не предполагает детальных знаний о принципах ее работы с целью модификации программы пользователем. Подобные модификации обычно проводят разработчики данной программной системы по заказу пользовательских организаций или специалисты по сопровождению. Такие сведения, как блок-схемы работы отдельных модулей системы на нижнем уровне детализации, описание принципов их работы и связей между ними включаются в документацию по усмотрению разработчиков.

Мы рассмотрели некоторые организационно-технические вопросы разработки программных систем. Естественно, что в каждом случае варианты создания систем могут быть намечены лишь с учетом конкретной ситуации, которая определяется рядом факторов: сферой применения и характером решаемых задач, квалификацией разработчиков и потенциальных пользователей, затратами и сроками проведения разработки и др. Таким образом, в создании программной системы проявляются особенности и свойства крупной сложной технической разработки. Поэтому вопросы организации и проведения таких разработок должны быть в каждом случае детально изучены и проанализированы с позиций реализации комплекса научно-исследовательских и опытно-конструкторских работ.

3. 4. ВНЕДРЕНИЕ И СОПРОВОЖДЕНИЕ ПРОГРАММНЫХ СИСТЕМ

Как и всякая сложная техническая система, программная система после разработки должна пройти определенную стадию внедрения у пользователей, включающих ее в свою систему обработки данных. В процессе работы программа должна подвергаться систематическому контролю и анализу. Частично данную работу могут проводить разработчики и специалисты по сопровождению, однако некоторые функции выполняются только в процессе эксплуатации программ пользователями. Рассмотрим задачи, возникающие на этом этапе.

Установка программной системы. Под установкой программной системы понимается подготовка ее к работе в условиях конкретного применения. Важнейшие задачи этого этапа, связанные с настройкой программы, описаны в документе «Руководство системного программиста» и частично — в документе «Руководство программиста».

В зависимости от сложности системы эти функции должны выполняться специалистами по сопровождению или программистами-пользователями. Так, если требуется генерация программной системы, желательно участие в этой работе разработчиков или специалистов по сопровождению. Обычно задача привязки для методо-ориентированных программных систем решается проще, чем для проблемно-ориентированных, в силу большей сложности последних и более жесткой их связи с конкретными прикладными задачами пользователей. Для большинства проблемно-ориентированных систем этап установки программы не ограничивается осуществлением ее привязки. Так, программные системы для решения задач управления обычно требуют организации массивов данных (в частности, определения форматов исходных данных, типов носителей информации, которые будут использованы, и форматов данных в основных массивах). При организации данных необходимо учитывать особенности программной системы и рекомендации по эффективной организации данных, приведенные в документации. Подготовку данных для организации массивов следует

вести в соответствии с правилами применения языковых средств системы. Если массивы для работы программы будут находиться в банке данных, организацию ввода и создания основных массивов выполняют программы системы управления базой данных. В ином случае для этой цели могут быть предусмотрены специальные модули в самой программной системе.

Другой задачей, решаемой в процессе установки программной системы в интегрированной системе обработки данных, является увязка данной программы с уже имеющимися. Обычно предполагается совместное использование различными программными системами основных массивов исходной информации и результатов обработки. С этой точки зрения все программные системы, кроме системы управления базой данных, по отношению к рассматриваемой программе могут быть условно разделены на несколько групп: программы, использующие для своей работы данные, общие с применяемыми в рассматриваемой системе; программы, использующие результаты работы системы; программы, результаты работы которых используются системой; программы, не связанные с рассматриваемой системой по исходным данным и результатам. Если программа не разрабатывалась специально для конкретной системы обработки данных, в первых трех случаях необходима доработка СОД для того, чтобы избежать дублирования данных и одновременно обеспечить обращение указанных программных систем к одним и тем же массивам в базе данных.

Сложность доработки обусловлена тем, что форматы данных и организация массивов, принятые в программной системе, могут не соответствовать организации данных в базе. Более того, в некоторых случаях информация, содержащаяся в базе данных, вообще не может быть использована непосредственно, однако, данные, необходимые для работы программы, можно получить из имеющихся несложными арифметическими и логическими преобразованиями. При наличии в интегрированной системе банка данных такая привязка обеспечивается с помощью вспомогательных программ, при разработке которых широко применяются языковые средства системы управления базой данных. Вспомогательные программы должны быть разработаны с учетом их применения в составе той или иной программной системы. Если новая программа использует ту же информацию, что и уже разработанная ранее программа, а также в случае, когда она использует результат работы какой-либо другой программы, иногда бывает удобнее модифицировать эту ранее разработанную программу.

Привязка системы к требованиям конкретного применения может вызвать необходимость внесения каких-либо изменений в программы системы. При этом обычно можно ограничиться модификацией нескольких программных модулей или добавлением в систему новых программ. Такая работа выполняется разработчиками или специалистами по сопровождению совместно со специалистами пользовательской организации. Если программная система относительно проста и к ней дается расширенная эксплуатационная до-

кументация, некоторые модификации пользователи могут провести самостоятельно. Однако необходимая для этого квалификация существенно превосходит тот уровень, который требуется для эксплуатации системы.

Обучение обслуживающего персонала и пользователей. Хотя эксплуатация программных систем упрощается по сравнению с ранее использовавшимися программами, этот вопрос требует специального рассмотрения, так как одновременно расширяется круг пользователей, применяющих подобные системы. Новые пользователи нуждаются в определенных сведениях о вычислительных машинах вообще и эксплуатации прикладных программ в частности. Кроме того, в некоторых случаях сложность самой программной системы проявляется в процессе эксплуатации. Здесь требуется обучение не только пользователей системы, но и обслуживающего персонала, работающего с подобной программной системой. Это, например, проявляется при эксплуатации банков данных. Здесь для обучения обслуживающего персонала может оказаться недостаточно только эксплуатационной документации. Опыт показывает, что для использования таких систем наряду с эксплуатационной документацией необходим выпуск учебных пособий и организация специальных курсов лекций, что позволит расширить круг пользователей системы и ускорит процесс ее внедрения.

Анализ системы в процессе эксплуатации. В процессе работы программной системы необходимо регулярно проводить ее анализ с целью определения производительности и оценки соответствия изменяющимся целям и требованиям пользователя. Для этого полезно регулярно учитывать время работы программы с тем или иным объемом данных и с различными вариантами запроса обработки. Это время можно сравнивать с нормативным временем, указанным в документации. Существенные расхождения здесь обуславливаются обычно ошибками при установке системы или в выборе способов ее эксплуатации при решении задач пользователями.

В процессе работы с программой могут выявиться ошибки в отдельных вариантах решения задачи. Подобные ошибки необходимо учитывать. В дальнейшем подробную информацию о них можно передать специалистам по сопровождению для устранения ошибок.

Важной характеристикой работы программной системы является ее гибкость в плане объединения с новыми системами, включаемыми в СОД. Кроме того, можно анализировать гибкость программы с позиций восприятия ею любых улучшений и модификаций в технических средствах и операционной системе ЭВМ. Если подобные улучшения трудно поддаются использованию, программа быстро устаревает морально и заменяется более эффективной.

Все сведения по анализу работы программы пользователи должны учитывать и передавать специалистам по сопровождению. Только таким образом можно организовать обратную связь от экс-

платации программы к ее доработке, которая необходима для создания качественной программной продукции.

Сопровождение прикладных программ. Опыт показывает, что разработка программной системы не заканчивается ее отладкой, созданием документации и распространением. На основе информации, поступающей от пользователей, необходимо постоянно дорабатывать систему, устраняя выявленные ошибки и совершенствуя эксплуатационные характеристики программы. При этом сведения об изменениях, внесенных в систему, должны сообщаться всем пользователям, особенно если это связано с исправлением ошибок, выявленных в ее работе. Важность этой деятельности становится очевидной, если учесть огромные объемы затрат на разработку прикладных программ и их эксплуатацию.

Таким образом, организации, занятые разработкой прикладного программного обеспечения, и пользователи должны иметь специальную централизованную службу сопровождения, которая бы учитывала информацию о распределении той или иной программной продукции, об изменениях в ней и т. п. Эта служба призвана практически реализовать связь между разработчиками и пользователями, необходимую для нормального развития системы прикладного программного обеспечения в целом.

Глава 4

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ СПУ/ЕС КАК ПРИМЕР ПРОГРАММНОЙ СИСТЕМЫ

В качестве примера прикладной программы, имеющей архитектуру программной системы, в этой главе рассматривается пакет прикладных программ для решения задач сетевого планирования и управления (ППП СПУ/ЕС). Выбор данной программной системы для иллюстрации материала предыдущих глав обусловлен целым рядом соображений.

ППП СПУ/ЕС представляется типичной программной системой, содержащей большинство элементов, присущих программам данной архитектуры. В связи с этим на примере пакета СПУ/ЕС имеется возможность конкретизировать некоторые положения, изложенные выше. Поскольку описание программной системы невозможно без анализа модели, которую реализует программа, объем изложения может существенно увеличиться за счет описания математических методов, не связанных непосредственно с вопросами программной реализации. Методы сетевого планирования в этом смысле выгодно отличаются простотой и наглядностью основных идей, а также широкой распространенностью сетевых моделей при решении определенного класса задач в системах управления разрабатываемыми предприятиями, при управлении строительными и ремонтными работами и т. д. Несмотря на относительную простоту (или благодаря ей), сетевые методы используются при решении одного из наиболее важных классов задач в автоматизированных системах управления разрабатываемыми предприятиями — задач оперативного управления научными исследованиями и опытно-конструкторскими разработками. В связи с этим описание программной реализации модели СПУ на базе ЕС ЭВМ может иметь и достаточно большое самостоятельное значение.

4. 1. ОСНОВНЫЕ СВЕДЕНИЯ О СЕТЕВЫХ МЕТОДАХ ПЛАНИРОВАНИЯ И УПРАВЛЕНИЯ

4. 1. 1. Управление разработками сложных систем

В условиях научно-технической революции существенно повышается значение научно-исследовательских и опытно-конструкторских работ. Наряду с расширением объемов этих работ можно отметить усложнение самих разрабатываемых изделий. Следствием этого является развитие специализации подразделений, организаций и предприятий, а также кооперации их при создании той или иной технической системы. Развитие кооперации усложняет задачи управления разработкой, так как при этом увеличивается число управляемых единиц (подразделений, организаций и предприятий различных уровней) и связей между ними. Кроме того, при усложнении разрабатываемых изделий увеличивается также объем материальных затрат и ресурсов, необходимых для их создания.

В отличие от других форм научно-исследовательской и производственной деятельности разработка сложных систем имеет определенные особенности, которые проявляются и при решении задач управления.

Определенность и уникальность разработки. Всякая разработка имеет определенную конечную цель, которая обычно состоит в создании нового изделия. При этом, как правило, ее проведение ограничено как сроками, так и объемом затрат. Технические характеристики изделия и организационно-экономические аспекты разработки в большинстве случаев задаются вначале и определяют условия создания конечного продукта — опытного образца и технической документации на разработанное изделие. Само изделие является новым в смысле отсутствия прямых аналогов и неповторяемости процесса разработки.

Новизна изделия не всегда связана с принципиально новыми методами его конструирования и новой технологией производства. Разработка сложной технической системы включает выполнение большого объема работ, по многим из которых уже существует опыт их проведения. Однако в процессе создания нового изделия всегда имеются работы, по которым такой опыт либо достаточно мал, либо вообще отсутствует. Чем больше объем таких работ, тем сложнее сама разработка, тем выше неопределенность в ее проведении, а следовательно, сложнее задача управления.

После того как разработанное изделие поступает в серийное производство, организации, которые были заняты его проведением, переходят к реализации следующей разработки, отличающейся как техническими характеристиками изделия, так и организационно-экономическими аспектами проведения разработки. В этом плане изготовление опытного образца изделия отличается от серийного производства изделия в условиях завода, где этапы производственного процесса повторяются для получения одних и тех же результатов.

Комплексность разработки. Необходимо отметить организационно-техническую сложность проведения разработки. Практически в процессе создания нового изделия проявляются все те производственные связи, которые в дальнейшем будут наблюдаться в процессе серийного выпуска изделия. Однако помимо них разработка содержит ряд уникальных компонентов, которые на этапе производства уже выпадают. Таким образом, сложность работ и их взаимосвязей в процессе разработки изделия значительно выше, чем в процессе его серийного выпуска. При этом увеличивается сложность планирования и управления разработкой, а также повышаются требования к ним. Планирование разработки должно обеспечить эффективное взаимодействие больших коллективов специалистов и возможности как последовательного, так и параллельного (где это допускается) выполнения работ, а также параллельного использования ресурсов и средств для различных работ. В результате этого обеспечивается выполнение организационно-экономических условий проведения разработки.

4. 1. 2. Методы сетевого планирования и управления

Эффективное управление разработкой требует постоянного контроля за временем и ресурсами, выделенными на выполнение работ. Для решения возникающих при этом задач применяются различные методы управления. Среди существующих в настоящее время методов управления крупными научно-техническими работками особое место занимают те, которые используют понятие сети для моделирования всего комплекса работ по созданию нового изделия [57]. Под моделью здесь понимается ряд методов и средств, позволяющих наглядно отобразить план проведения разработки, определить основные ее характеристики, используемые в процессе управления, а также сравнить альтернативные варианты возможных решений тех или иных задач без их практической реализации.

Отметим, что методы сетевого планирования и управления могут применяться не только для проведения научно-исследовательских и опытно-конструкторских работ, но также в строительстве, при ремонтах крупных сооружений, планировании военных операций и т. п. В частности, в ходе изложения применение этих методов иллюстрируется на примере задачи установки и наладки ЭВМ на вычислительном центре. При этом будут описаны основные этапы построения сетевых графиков и работы с ними.

Составление перечня работ. Первым этапом планирования разработки всегда в той или иной форме является разделение всего проекта на отдельные работы, которые должны быть выполнены для его реализации. При этом должны быть заранее определены степени детализации работ, необходимые для эффективного управления на различных уровнях. Так, в процессе составления плана установки ЭВМ подготовка машинного зала может рас-

смагиваться как работа, хотя она не является однородной и может состоять из подготовки специальных полов, вентиляционных приспособлений и т. п. При другом уровне детализации каждая из перечисленных работ может фигурировать в плане отдельно.

Критериями при выборе степени детализации работ являются обычно время, необходимое для выполнения работы, средние за-

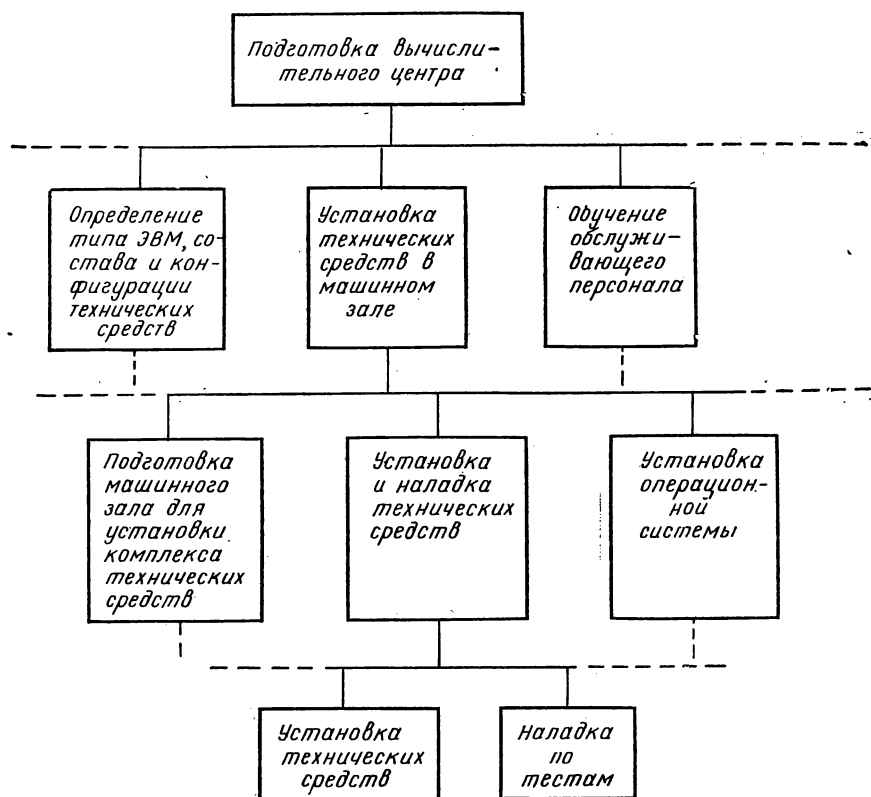


Рис. 24. Фрагмент перечня работ проекта подготовки вычислительного центра

траты на ее выполнение, уровень руководства, ответственного за проведение работ, и др.

Простой метод составления перечня работ, описываемый ниже, может служить гарантией его полноты. Вначале вся разработка разбивается на несколько крупных работ, выполнение которых заведомо обеспечивает реализацию проекта. Затем каждая из укрупненных работ в свою очередь разбивается на более детальные работы с учетом полноты этой разбивки. Продолжая этот процесс, приходим к необходимой степени детализации, не утратив полноты в процессе разбивки, какой бы мелкой она ни была. На рис. 24

показан фрагмент примерного перечня работ по организации вычислительного центра.

Некоторые из работ составленного перечня должны выполняться в четко определенной последовательности — одна за другой, другие — параллельно (одновременно или с перекрытием). Возможность параллельного выполнения работ является основой для сокращения времени на проведение всей разработки. В связи с этим учет подобной возможности представляет основную задачу в процессе планирования.

Составленный перечень работ является лишь основой плана. В нем не отражаются наглядно взаимосвязи между отдельными работами. Для удобства планирования и дальнейшего управления ходом разработки используются специальные схемы — сетевые графики, которые в некоторой условной форме представляют план реализации проекта.

Правила построения сетевых графиков. Как и всякое условное изображение (чертеж детали, принципиальная схема прибора и пр.), сетевые графики необходимо строить по определенным правилам, обеспечивающим однозначное отображение плана разработки, а также однозначное восстановление плана по имеющемуся графику.

Работы проекта изображаются на графике стрелками. Начальная точка стрелки называется начальным (или предшествующим) событием данной работы, конечная точка — конечным (или последующим) событием. События на графике обычно выделяются кружками (рис. 25). Каждое событие идентифицируется определенным буквенно-цифровым символом, называемым именем или номером события. Различные события в сетевом графике должны иметь разные имена.

Работа в сети идентифицируется совокупностью имен предшествующего и последующего событий, стоящих в фиксированном порядке: «имя предшествующего события» — «имя последующего события». Например на рис. 25, а изображен сетевой график, в котором имеются работы 1—3, 3—4, но нет работ 2—5 или 5—3.

Если какие-либо две работы должны быть выполнены последовательно, то на графике конечное событие первой работы будет являться начальным для второй. Так, на рис. 25, б работа 1—3 предшествует работе 3—4. Выполнение последней не может начаться раньше окончания работы 1—3.

Само событие является моментом в выполнении плана и характеризуется в этом смысле датой наступления. Отметим, что событие не представляет в разработке отрезок времени, а является точкой на временной линии осуществления разработки.

Говорят, что событие свершилось или наступило, если выполнены все работы, для которых данное событие является конечным (или, иначе, все работы, «входящие» в данное событие). Работа может быть начата только тогда, когда наступает ее начальное событие, т. е. событие, из которого она выходит. Совокупность всех работ разработки (с учетом заданного уровня детализации), изоб-

раженных в виде стрелок, и промежуточных событий, связывающих их, называется сетевым графиком разработки, или сетью.

Сетевой график имеет одно или несколько событий, называемых начальными событиями сети. Начальные события характеризуются тем, что в них не входит ни одной стрелки. Эти события изображают точки начала проекта. Аналогично сетевой график имеет одно или несколько событий, называемых конечными событиями сети. Они характеризуются тем, что из них не выходит ни одной стрелки. Эти события изображают точки окончания проекта*.

Все остальные события графика, не определенные как начальные или конечные для сети, должны обязательно иметь входящие и выходящие стрелки. В противном случае говорят, что график имеет соответственно висящие начальные или висящие конечные события и это рассматривается как ошибка при его составлении.

Предположим, что в процессе составления сетевого графика мы столкнулись с ситуацией, показанной на рис. 25а. «Прочесть» этот фрагмент сети нужно следующим образом: работы 3—4 и 3—5 могут быть начаты лишь после окончания работ 1—3 и 2—3. Однако на самом деле может оказаться, что лишь работа 3—4 зависит от работ 1—3 и 2—3, а работа 3—5 зависит только от работы 2—3. Для отображения подобной ситуации во избежание разночтений вводится так называемая фиктивная работа или стрелка, изображенная на рис. 25, б. Фиктивная стрелка не описывает никакой реальной работы, требующей для своего выполнения определенного времени и ресурсов. Она указывает лишь на логическую взаимосвязь событий. Так, событие 3 на рис. 25, б не может свершиться раньше события 3'. Таким образом, с введением фиктивной работы график на рис. 25, а приобретает однозначное толкование, приведенное в начале.

Хотя в процессе проведения разработки могут встретиться ситуации, когда определенная серия работ должна быть повторена более одного раза (например, результаты последующих работ выявили необходимость переделки результатов некоторой предшествующей работы) и не всегда такие случаи можно предсказать заранее, в сетевом графике они должны быть по возможности учтены. Это вызвано тем, что методы обработки сетевых графиков не допускают существования в сети замкнутых путей, называемых петлями (рис. 26).

Приведенные правила обеспечивают составление сетевых графиков, полностью соответствующих плану разработки. При этом сеть имеет существенное преимущество перед традиционным планом — наглядность. Составление сетевого графика разработки представляет собой второй этап работы по планированию с применением сетевых методов. При этом сетевой график базируется на перечне работ, выполненном на первом этапе планирования.

* Следует различать понятия «начальные» и «конечные» события для сети в целом и для работ сетевого графика.

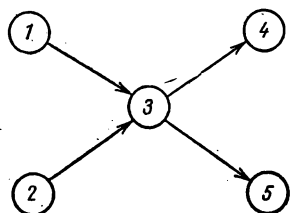
Обычно для разработки создается одна сеть. Однако единый проект может разбиваться на ряд определенных частных проектов, обусловленных требованиями всей разработки. В этом случае каждый из таких частных проектов может описываться собственной сетью. Сеть для всей разработки в этом случае можно рассматривать как совокупность сетей для частных проектов. В дальнейшем сети для частных разработок будем называть подсетями.

В процессе разработки сети определяются события, в которых подсети связываются друг с другом. В дальнейшем программа использует события-связи для объединения подсетей и обработки их как единой сети. Для этой цели каждое событие-связь имеет одно и то же имя во всех подсетях, в которые оно входит. Различные события-связи имеют разные имена.

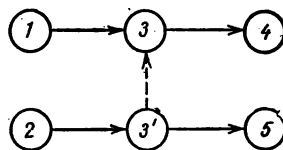
Подсеть может быть обособленной единицей разработки или связанной с другими подсетями в сети. Если подсеть является обособленной, то она не сопрягается с другими подсетями в сети, т. е. не имеет событий-связей (рис. 27). Например, при составлении сетевого графика установки и внедрения ЭВМ подсеть, описывающая план подготовки персонала для обслуживания вычислительных средств, может быть обособленной. С другой стороны, подсети подготовки машинного зала, установки и наладки оборудования должны быть связаны.

Оценка времени выполнения работ. Следующим этапом планирования является оценка времени, необходимого для выполнения каждой работы. Для любой работы может быть задана одна временная оценка, называемая ожидаемым временем (обозначим ожидаемое время t_0) выполнения работы, которая используется при обработке сетевых графиков.

Другим возможным вариантом оценки ожидаемого времени, который в некоторой степени учитывает неопределенность наших знаний о действительно необходимом времени, является метод трех временных оценок. В этом случае задаются три оценки времени выполнения работы: оптимистическая ($t_{оп}$), наиболее вероятная ($t_{нв}$) и пессимистическая ($t_{п}$). Под оптимистической оценкой времени понимается минимальное время, требуемое для выполнения работы. Она определяет случай, когда работа производится без каких-либо неожиданных ситуаций. Этот результат обычно не достигается. Оценка наиболее вероятного времени представляет нормальное время, требуемое для выполнения работы. Этот результат мог бы иметь место наиболее часто, если бы одна и та же работа была повторена несколько раз без учета предыдущего



а



б

Рис. 25. Введение фиктивной стрелки для однозначного описания плана с помощью сетевого графика

опыта. Пессимистическая оценка времени — это оценка максимального времени выполнения работы, т. е. результат, который может наблюдаться при исключительно неблагоприятных условиях. Эти три оценки обрабатываются простыми статистическими методами,

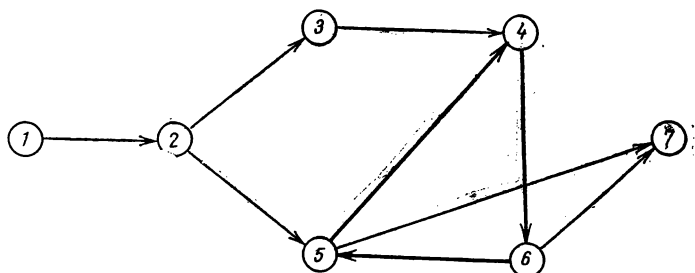


Рис. 26. Сетевой график, содержащий петлю

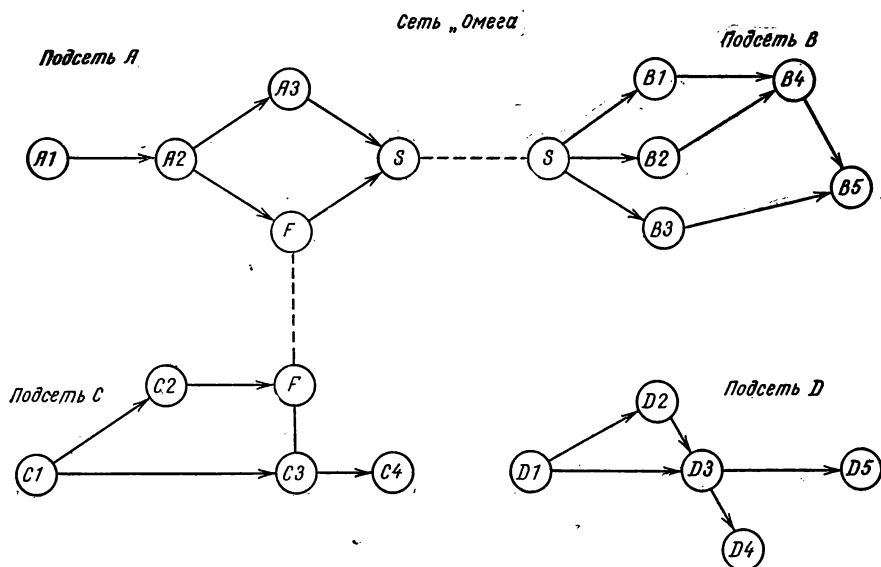


Рис. 27. Варианты взаимосвязи подсетей в сети «Омега»: подсети А, В и С связаны, подсеть D — обособленная

в результате чего получается некоторая единственная оценка, используемая в дальнейших расчетах как ожидаемое время (t_0), необходимое для проведения этой работы. Математические методы определения оценки времени с учетом ряда предположений дают следующую формулу для нахождения ожидаемого времени по трем оценкам:

$$t_0 = \frac{t_{\text{оп}} + 4t_{\text{нв}} + t_{\text{п}}}{6} \quad (1)$$

Оценки времени выполнения работ могут быть получены либо путем использования имеющихся нормативов, либо (если работа ранее не выполнялась) могут быть применены экспертные оценки.

Расчет временных характеристик сети. После построения сети и оценки времени выполнения каждой работы можно приступить к расчету временных характеристик сети, имеющих важное значение в процессе управления разработкой.

На рис. 28 показана сеть, где стрелками указаны работы, а кружки с буквами идентифицируют события. Числа над стрелками указывают продолжительность соответствующих работ, выраженную в неделях. Если работы в сети начинаются, например, 7 апреля 1975 г., то дата завершения работы С—D будет 28 апреля 1975 г. Эта дата вычисляется путем суммирования ожидаемых периодов времени (t_0) для работы А—В, В—С и С—D (три недели) и прибавления этого времени к заданной дате начала — дате наступления события А. Аналогично дата завершения работы В—D — 7 мая 1975 г. (с учетом двух праздничных дней). Эти две работы (С—D и В—D) заканчиваются одним и тем же событием D. Одно из правил построения и чтения сетевых графиков гласит, что событие наступило, если завершены все работы, для которых данное событие является конечным. Поэтому ожидаемая дата (T_0) свершения события D — 7 мая 1975 г. Таким же образом вычисляются ожидаемые даты свершения других событий в сети. В частности событие F имеет дату наступления — 29 мая 1975 г. (с учетом праздничных дней), а событие G — 5 июня 1975 г. Отсю-

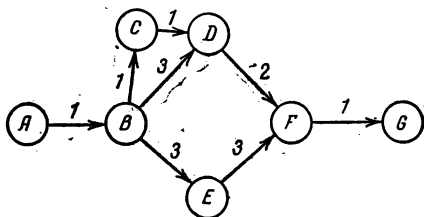


Рис. 28. Сеть с заданными для расчета оценками времени выполнения работ

да получаем ожидаемую дату завершения всей разработки и ее продолжительность. Кроме того, можно определить ту цепочку работ, суммирование оценок времени для которых дано время выполнения всего проекта. В данном примере такую цепочку (или путь) составляют работы А—В, В—Е, Е—F, F—G. Они образуют так называемый критический путь, представляющий важную характеристику проекта. С нахождением критического пути выделяются те работы в сети, которым должно быть уделено наибольшее внимание в процессе выполнения разработки. Отсрочка выполнения любой из работ критического пути по сравнению с данной оценкой автоматически вызывает срыв срока наступления одного из конечных событий разработки. Если сеть содержит только одно начальное и одно конечное событие, то этот путь является самым длительным по времени путем в сети и определяет длительность всей разработки. В случае наличия в сети нескольких начальных и конечных событий различные критические пути могут иметь разную длительность.

Общая продолжительность работ в сети может быть также ограничена директивной (плановой) датой наступления конечного события сети (или несколькими плановыми датами, если имеется несколько конечных событий). Если событие G (см. рис. 28) имеет плановую дату ($T_{пл}$) 3 июня 1975 г., то можно вычислить самую позднюю дату наступления события D ($T_{сп}$), обеспечивающую выполнение установленного срока — 13 мая 1975 г. Эта дата вычисляется суммированием оценок времени для работ D—F и F—G и вычитанием результата из плановой даты события G. Аналогично может быть найдена самая поздняя дата наступления любого другого события сети. Самая последняя (самая поздняя) допустимая дата ($T_{сп}$) есть такая дата, до которой событие должно произойти, чтобы плановая дата конечного события не была сорвана. Если плановая дата для конечного события не задана, то в этом случае ее роль выполняет ожидаемая дата наступления этого события.

Другими важными характеристиками разработки являются резервы времени выполнения работы. Первичный резерв времени ($t_{пр}$) выполнения работы рассчитывается по следующей формуле:

$$t_{пр} = T_{спк} - T_{он} - t_0, \quad (2)$$

где $T_{спк}$ — самая поздняя допустимая дата наступления конечного (последующего) события данной работы;

$T_{он}$ — ожидаемая дата наступления начального (предшествующего) события;

t_0 — оценка времени выполнения работы.

Так, первичный резерв работы D—F на рис. 28 равен 0,6 недели (при пятидневной неделе).

Первичный резерв характеризует интервал времени, на который может быть отсрочено выполнение работы в сети без срыва конечного срока реализации проекта. Отсюда следует, что работы критического пути имеют минимальный первичный резерв. Отметим, что полное использование первичного резерва работы приводит к наступлению последующего события в самое позднее допустимое время. Это означает, что хотя бы один из путей, соединяющих данное событие с конечным событием сети, становится критическим. Таким образом, в дальнейшем, после выполнения данной работы, число критических путей, а значит и критических работ в сети увеличивается, что сокращает возможности эффективного управления ходом разработки.

Плановая дата может быть задана и для промежуточных работ и событий. В этом случае резерв времени выполнения работ, предшествующих данному событию, может быть вычислен с учетом этой плановой даты. Величина, полученная как значение наиболее критичного резерва из всех имеющихся у данной работы (вычисленных относительно различных плановых дат), называется вторичным резервом. Вторичный резерв в этом смысле представляет еще одну временную характеристику работы. Так, первичный ре-

зерв работы С — D на рис. 28 равен 1,6 недели. В то же время вторичный резерв для этой работы может быть вычислен от ожидаемой даты наступления события D. В этом случае вторичный резерв работы С — D равен одной неделе.

Вторичный резерв, исчисляемый от ожидаемой даты наступления последующего события, с позиций управления является более удобной характеристикой работы, чем первичный резерв. Использование вторичного резерва времени выполнения работы в меньшей степени влияет на весь дальнейший ход выполнения проекта, поскольку при этом не отодвигается ожидаемая дата наступления последующего события.

Часто удобнее бывает вычислять дату окончания работы, а не наступления события. Аналогично предыдущим определениям можно ввести величины, характеризующие самую позднюю дату ($T_{спр}$) и ожидаемую дату выполнения работы ($T_{ор}$). При этом формула расчета первичного резерва времени выполнения работы будет следующей:

$$t_{пр} = T_{спр} - T_{ор} \quad (3)$$

В табл. 3 с учетом этого замечания приведены основные характеристики работ сетевого графика, показанного на рис. 28.

Таблица 3

ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ РАБОТ СЕТЕВОГО ГРАФИКА

Работа	Оценка ожидаемого времени выполнения работы (t_0), неделя	Ожидаемая дата окончания работы ($T_{ор}$)	Самая поздняя допустимая дата окончания работы ($T_{спр}$)	Первичный резерв времени выполнения работы ($t_{пр}$), неделя	Вторичный резерв времени, исчисляемый от наиболее критичной даты наступления последующего события, неделя
А—В	1	14 апр. 1975 г.	12 апр. 1975 г.	—0,4	—0,4
В—С	1	21 апр. 1975 г.	5 мая 1975 г.	1,6	1
В—D	3	7 мая 1975 г.	13 мая 1975 г.	0,6	0
В—Е	3	7 мая 1975 г.	5 мая 1975 г.	—0,4	—0,4
С—D	1	28 апр. 1975 г.	13 мая 1975 г.	1,6	1
D—F	2	22 мая 1975 г.	27 мая 1975 г.	0,6	0
Е—F	3	29 мая 1975 г.	27 мая 1975 г.	—0,4	—0,4
F—G	1	5 июня 1975 г.	3 июня 1975 г.	—0,4	—0,4

Начиная с 50-х годов методы управления, основанные на использовании сетевых графиков, нашли широкое применение в различных системах управления разработками сложных систем. В нашей стране эти методы получили наибольшую известность под названием «сетевое планирование и управление» (СПУ). Сетевое

планирование и управление можно определить как систему принципов и методов эффективного планирования хода разработки на основе построения и сопровождения сетевых графиков, устанавливающую базу для расчета основных характеристик плана разработки, его контроля и корректировки. В данном параграфе была рассмотрена составная часть системы СПУ (называемая обычно СПУ — ВРЕМЯ), обеспечивающая возможность планирования и контроля за ходом разработок по временным характеристикам плана.

Следует отметить, что ППП СПУ/ЕС реализует гораздо более развитый набор методов сетевого планирования, нежели тот, который изложен выше. В частности, здесь не описаны методы расчета затрат и ресурсов, связанных с выполнением разработки. Однако для иллюстрации принципов работы программы, имеющей архитектуру программной системы, в данной книге рассматривается только программная реализация методов расчета временных характеристик сетей.

Одним из основных требований, предъявляемых к прикладным программам в АСУ, является простота эксплуатации таких программ. Для иллюстрации этого положения ниже подробно описываются средства ППП СПУ/ЕС, обеспечивающие удобство применения пакета для пользователей.

4. 2. ВОЗМОЖНОСТИ ППП СПУ/ЕС И ЕГО ТРЕБОВАНИЯ К ВЫЧИСЛИТЕЛЬНЫМ СРЕДСТВАМ

4. 2. 1. Характеристики обрабатываемых сетей

Прежде чем включить ту или иную прикладную программу в собственную систему обработки данных, пользователь должен убедиться в том, что данная программа соответствует его требованиям к решению того круга задач, для которых она предназначена. С другой стороны, любая программа для своей работы предполагает определенный состав вычислительных средств, включая комплекс технических средств ЭВМ, состав операционной системы и, возможно, какие-либо дополнительные средства. Перед установкой программы пользователь должен убедиться в том, что его вычислительная система содержит все необходимые средства. Таким образом, на первом этапе работ по внедрению прикладной программы необходимо ознакомиться с ее основными особенностями и характеристиками.

Для решения задач СПУ в нашей стране и за рубежом существует значительное количество различных программ, предназначенных для разных вычислительных машин. Большинство простых программ дают пользователю возможность вводить лишь ограниченное количество элементов данных по каждой работе в рамках записей, имеющих жесткий формат. Даже более сложные программы, представляющие большие возможности по вводу и обработке

данных, не всегда удовлетворяют требованиям тех конкретных ситуаций, которые возникают в процессе управления сложными разработками. Часто эти требования могут быть удовлетворены путем модификаций программы, но модификация большой и сложной программы, не обеспеченной специальными средствами внесения изменений, требует больших затрат времени и средств.

ППП СПУ/ЕС представляет одно из возможных решений этой проблемы. С каждой работой в сети здесь может быть связано большое число различных элементов данных. Программа может организовать эти данные, обработать их и обеспечить формы выходных документов в соответствии с требованиями пользователей. Модульная структура системы упрощает задачу модификации программы. Модификация может осуществляться любым из тех способов, которые описывались выше для программных систем, за исключением генерации, которая для ППП СПУ/ЕС не требуется. Рассмотрим некоторые основные характеристики ППП СПУ/ЕС в плане обработки сетевых графиков.

Сети и подсети. Сетевые графики, обрабатываемые программой, могут быть либо едиными, либо расчлененными на подсети. Любая подсеть может быть как обособленной, так и связанной с другими подсетями в сети. Каждая подсеть может иметь несколько начальных и конечных событий. Общее число подсетей в сети, которое допускается программой, — около 250.

Размер сети, которая может быть обработана программой, зависит от числа работ в каждой подсети. Максимальное допустимое число работ в подсети колеблется (в зависимости от объема оперативной и внешней памяти вычислительной системы пользователя) от 1000 работ при объеме оперативной памяти в 44К до 32 000 работ при 1004К (без учета требований операционной системы).

Работы и события. Для любой работы может быть использована одна или три оценки времени. При этом оценки времени могут быть как целыми, так и дробными единицами времени. За единицу времени могут быть приняты: час, день, неделя, месяц, квартал или любой другой отрезок времени, например 10 дней. При необходимости работы в сети можно пометить как непрерывные. В этом случае при расчете временных характеристик будет учитываться, что работа не прерывается в выходные и праздничные дни, а выполняется непрерывно со дня начала до ее полного окончания.

С каждой работой в сети может быть связано большое число элементов данных: словесное описание работы, имя подсети, к которой относится работа, плановая дата завершения работы и др. Кроме того, пользователь имеет специальные языковые средства, с помощью которых он может добавить новые элементы данных или упразднить некоторые из существующих. Имена событий могут быть произвольными буквенно-цифровыми кодами и следовать в произвольном порядке. Начальным событиям сети присваиваются даты начала работ. Любым промежуточным и конечным собы-

тиям могут быть присвоены плановые даты наступления, которые также будут использованы при расчетах.

Генерация календаря. Для вычисления календарных дат начала и окончания работ программа обеспечивает расчет (генерацию) календаря с максимальным временным охватом — 15 лет для одной сети. Пользователь может указать период, на который ему необходим календарь, и программа проведет его генерацию.

Помимо субботних и воскресных дней, а также традиционных праздников пользователь может указать дополнительные праздничные дни. С другой стороны, существующие выходные дни могут быть идентифицированы как будние.

4. 2. 2. Возможности обработки сетевых графиков

Программа может обрабатывать как всю сеть в целом, так и каждую подсеть в отдельности, даже в случае, когда подсети связаны между собой. Если подсеть обрабатывается как связанная, программа учитывает при проведении расчетов влияние всех связанных с ней подсетей. Например, на рис. 27 самое раннее начало работы $F—C3$ ограничивается не только работой $C2—F$ в подсети C , но и работой $A2—F$ в подсети A . Если подсеть C обрабатывается как обособленная*, все связи ее с другими подсетями игнорируются программой. В этом случае работа $F—C3$ ограничена только работой $C2—F$ в своей подсети.

ППП СПУ/ЕС позволяет обрабатывать информацию о проводимых, но еще не законченных работах сети, а также учитывать фактическую информацию об окончании работ.

К обработке допускаются сети, в которых между двумя событиями может быть более одной работы. Такие работы должны идентифицироваться помимо кодов событий дополнительным символом. Программа обеспечивает расчет вторичного резерва для работ сетевого графика при заданных плановых датах наступления событий.

Укрупнение сетей. Программные реализации системы СПУ могут обеспечивать те или иные дополнительные возможности обработки сетевых графиков. В частности, одной из таких возможностей является укрупнение сети. Для выполнения многих больших проектов могут потребоваться на определенном уровне детализации тысячи и десятки тысяч работ. Анализ сетевого графика такого объема затруднителен, поскольку теряется наглядность в отображении плана. Поэтому желательно иметь возможность укрупнять сетевые графики, оставляя в них лишь важнейшие работы и события, по которым можно проследить основные моменты разви-

* Следует различать подсети, обособленные в сети (т. е. не связанные с другими подсетями), и обособленную обработку подсети, связанной с другими подсетями. В последнем случае подсеть обрабатывается как обособленная лишь в соответствии с запросом пользователя.

тия разработки. ППП СПУ/ЕС обеспечивает два типа укрупнения сетей: сжатие по границам и сжатие по работам.

При проведении сжатия по границам пользователь может идентифицировать некоторые события как «граничные». В этом случае укрупненная сеть будет содержать только начальные, конечные и

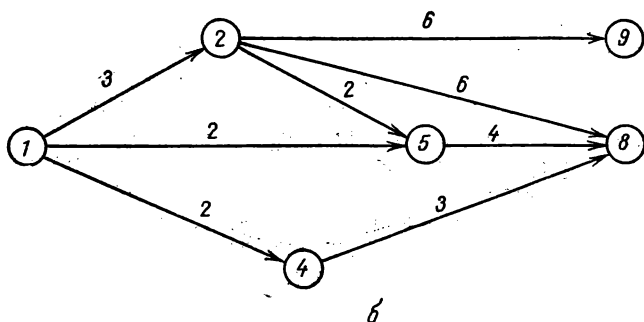
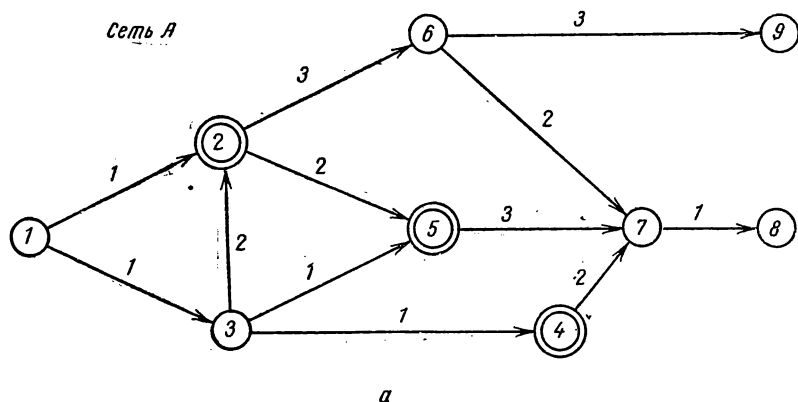


Рис. 29. Укрупнение сети по граничным событиям:
а — детальная сеть А, события 2, 4 и 5 — граничные; б — укрупненная сеть А

граничные события. На рис. 29, а события 2, 4 и 5 помечены как граничные. При сжатии сети А будет сформирована новая сеть (рис. 29, б). Эта укрупненная сеть содержит только начальные (1), конечные (8 и 9) и граничные (2, 4 и 5) события. Начальные и конечные события необходимы для сохранения структуры сети.

Между двумя событиями укрупненной сети работа имеется в том случае, когда в детальной сети эти события соединялись хотя бы одной цепью работ, не имеющей внутри ни одного граничного события. Так, работа 2—8 в укрупненном сетевом графике на рис. 29, б имеется потому, что существует путь 2—6, 6—7, 7—8 де-

тальной сети (рис. 29, а). Заметим, что путь 2—5, 5—7, 7—8 не определил бы работу 2—8, так как событие 5 — граничное.

Оценка времени выполнения работы 2—5 в укрупненной сети совпадает с оценкой времени ее выполнения в детальной сети: работа 1—5 укрупненного сетевого графика имеет продолжительность, равную сумме оценок времени выполнения работ 1—3 и 3—5 детальной сети. Аналогично вычисляются оценки времени для работы 1—4 (сумма оценок времени выполнения работ 1—3 и 3—4) и работы 4—8 (сумма оценок 4—7 и 7—8). Продолжительность выполнения работы 1—2 укрупненного графика будет определена на основании анализа двух возможных путей детальной сети: 1—2 (путь, состоящий из одной работы) и 1—3, 3—2. Выбирается наибольшая из двух полученных оценок (в данном случае по пути 1—3, 3—2 — 3 недели).

Процесс укрупнения полезен в тех случаях, когда требуется наглядно представить максимальную длительность путей между основными событиями. Очевидно, что приведенный метод расчета временных оценок в укрупненной сети обеспечивает совпадение длительностей критических путей в укрупненной и детальной сетях. Кроме того, выделяются те основные события, которые лежат на критических путях.

Следует, однако, отметить, что при укрупнении сети по граничным событиям отсутствует функциональная зависимость между числом детальных работ, числом работ в укрупненной сети и числом граничных событий. Это создает определенные трудности для использования укрупненной сети в управлении проектом.

Процесс укрупнения по работам исключает эти трудности, позволяя пользователю точно определять, какие работы он хочет иметь в укрупненной сети*. Эти работы, называемые агрегатными, пользователь определяет по их предшествующим и последующим событиям, которые идентифицируются как граничные. События могут принадлежать одной или двум различным подсетям (для каждой работы) и могут находиться (для нескольких работ), на связанном пути работ.

ППП СПУ/ЕС обеспечивает укрупнение и расчет основных временных характеристик укрупненной сети по запросу пользователя. Отметим, что обеспечивается несколько уровней укрупнения сети. Для этого граничное событие может помечаться как принадлежащее к определенному уровню. Как правило, уровень укрупнения может отражать уровни руководства разработкой. При этом если требуется укрупнение определенного уровня, например четвертого, то в сети останутся граничные события данного и более высоких уровней, т. е. 4, 3, 2 и 1-го (каскадные уровни). Программа позволяет работать с шестью каскадными уровнями укрупнения сетей, благодаря чему может быть достигнута именно та степень укрупнения графика, которая необходима в данный момент для данного

* Фактически имеется один общий метод укрупнения сети и два различных подхода к его использованию.

уровня руководства. Кроме того, программа дополнительно обеспечивает три обособленных (некаскадных) уровня укрупнения сетей.

4. 2. 3. Выходная информация

Формы выходных документов. ППП СПУ/ЕС обеспечивает пользователя большим числом стандартных форм отчетов по детальному и укрупненному сетевым графикам. Эти отчеты пользователь может затребовать, используя языковые средства программы. Кроме того, программа снабжена специальными языковыми средствами, с помощью которых пользователь может составить собственные, удобные ему формы выходных документов.

Контроль данных и запросов. Вся входная информация по сетевым графикам хранится в основном массиве (главном файле) и может быть при необходимости напечатана. Это обеспечивает возможность нахождения ошибок в исходных данных. Программа предоставляет также специальные средства для диагностики и редактирования ошибок в данных и запросах пользователя. При обнаружении ошибки пользователь может определить необходимость продолжения обработки, модифицировать производимую обработку или прекратить ее вообще. В последнем случае часто оказывается возможным после корректировки данных начать выполнение программы с того места, где была обнаружена ошибка.

4. 2. 4. Требуемая конфигурация вычислительных средств

Пакет прикладных программ СПУ/ЕС работает на ЭВМ Единой системы (ЕС-1020, ЕС-1030, ЕС-1040, ЕС-1050) под управлением операционной системы ОС ЕС с произвольной управляющей программой (РСР, MFT или MVT) [17]. Допустимые объемы обрабатываемых сетевых графиков в существенной мере зависят от имеющихся объемов оперативной и внешней памяти ЭВМ (табл. 4).

Теоретически минимальная конфигурация вычислительной системы, обеспечивающая расчет обособленной подсети объемом в 500 работ, составляет 64К оперативной памяти ЭВМ ЕС-1020 с одним устройством считывания с перфокарт, одним накопителем на магнитной ленте, алфавитно-цифровым печатающим устройством и двумя накопителями на магнитном диске [4]. Однако такая система не будет эффективной. В частности, она не обеспечит средства для обновления основного массива и при каждом выполнении программы этот массив должен будет возобновляться. Практический минимум, обеспечивающий расчет подсетей с числом работ около 1000, дает конфигурация, описанная выше, с добавлением одного накопителя на магнитных дисках и двух накопителей на магнитной ленте.

Поскольку ППП СПУ/ЕС широко использует последовательный метод доступа, существует взаимозаменяемость накопителей на магнитных дисках и лентах. Однако при работе программы с некоторыми массивами используется прямой метод доступа [17], и, следовательно, эти массивы могут быть расположены лишь на внешних устройствах с прямым доступом — магнитных дисках.

Эффективная конфигурация вычислительной системы зависит от объема обрабатываемых сетей и некоторых других параметров.

Таблица 4 :

ТРЕБОВАНИЯ ППП СПУ/ЕС К ОПЕРАТИВНОЙ И ВНЕШНЕЙ ПАМЯТИ*

Число работ в подсети	Требования к оперативной памяти, К байт	Требования к внешней памяти, К байт
1 000	44	700
3 560	108	2 492
8 680	236	6 076
18 920	492	13 244
32 000	1 004	22 400

* Здесь не учитываются требования операционной системы к оперативной и внешней памяти

ров. Документ «Руководство программиста» содержит формулы, позволяющие определить необходимую конфигурацию по этим параметрам.

4. 3. СТРУКТУРА И ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ ППП СПУ/ЕС

После принятия решения о включении прикладной программы в систему обработки данных необходимо ознакомить персонал, который будет занят ее эксплуатацией, с основными вопросами функционирования программы. При этом, естественно, различным категориям персонала требуются различные сведения о работе программ пакета. В этом параграфе мы не преследуем цели дать описание программы для различных категорий специалистов, а проиллюстрируем лишь важнейшие принципы работы программной системы на примере ППП СПУ/ЕС. Изложение содержит сведения, необходимые различным категориям обслуживающего персонала.

4. 3. 1. Принципы построения ППП СПУ/ЕС

Основным принципом построения пакета является развитая многоуровневая модульность, обеспечивающая те преимущества, которые предоставляются вычислительными системами третьего поколения. Преимуществом модульного метода является то, что продолжительность срока службы набора модулей значительно выше срока службы каждого из отдельных его элементов,

На самом верхнем уровне иерархии программа состоит из трех крупных библиотек модулей, называемых процессорами (следует отличать этот термин от термина, определяющего центральное вычислительное устройство ЭВМ):

- процессора обработки временных характеристик сетевых графов (процессор сетей, или сетевой процессор);

- процессора обработки данных по затратам на проведение разработки (процессор затрат);

- процессора формирования выходных документов (процессор отчетов).

Процессоры оформлены как обычные библиотеки программ пользователя и для операционной системы имеют статус личных библиотек [17]. Обращение к ним осуществляется с помощью управляющей карты //JOBLIB DD . . ., содержащей в параметре DSNAME имя данного процессора [17]. Например, управляющая карта описания процессора сетей может иметь вид

```
//JOBLIB DD DSNAME = SPU. PRSETEJ, UNIT = 5050*,  
VOL = SER = GRAFIK, DISP = (OLD, KEEP)
```

Как видно из названий процессоров, уже на самом высоком уровне модульности прослеживается систематическое разделение программ, реализующих определенные функции из состава задач, решаемых пакетом СПУ/ЕС. Так, в частности, выделение в пакете отдельного процессора для формирования отчетов обеспечивает необходимую гибкость программы, выражающуюся в возможности приспособления форм выходных документов к требованиям конкретного пользователя. Данный процессор является наиболее крупным и сложным по структуре компонентом в составе пакета, что в определенной мере служит характеристикой его широких возможностей в плане привязки программы к специфике задач пользователя.

Результаты работы программ процессора сетей или затрат не предназначены для непосредственной выдачи пользователю. Они оформляются в виде промежуточных массивов данных. Эти массивы предоставляют исходную информацию для программ процессора отчетов, которые и заканчивают обработку созданием форм выходных документов, затребованных пользователем. Практически поток информации в программе может реализоваться по одному из трех вариантов (рис. 30).

Каждый процессор содержит набор модулей, называемых процедурами и определяющих следующий уровень модульности пакета. Процедуры представляют собой основные программы, ведущие обработку данных пользователя. С точки зрения выполнения программы СПУ/ЕС как задания на вычислительной машине каждая процедура образует шаг задания [17]. Она вызывается из соответствующей библиотеки и подготавливается для выполнения по управляющей карте //...EXEC..., содержащей в параметре PGM имя данной процедуры [17].

Разделение функций на уровне процедур и особенности реализации этих функций поясним на примере процедур процессора сетей.

Сетевой процессор имеет в своем составе следующие процедуры (рис. 31):

- редактирования вводимых данных;
- упорядочения отредактированных данных;
- формирования основного массива данных;
- обработки сетевых графиков.

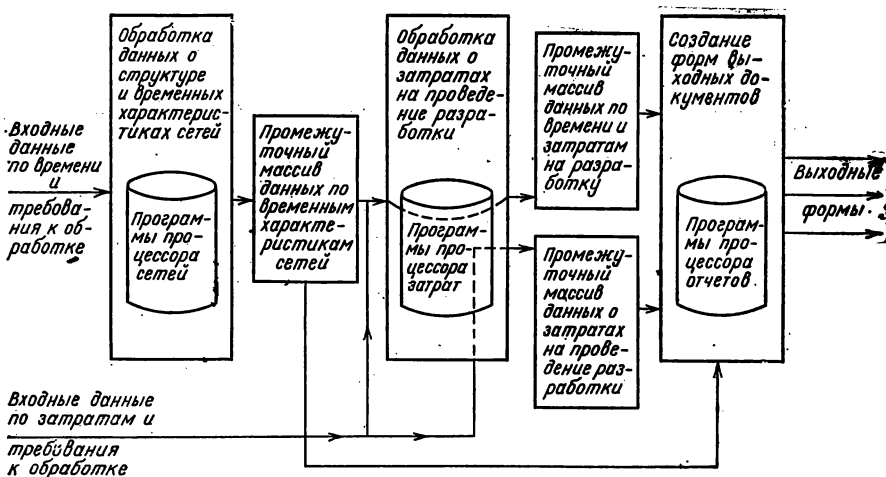


Рис. 30. Варианты реализации информационного потока в процессорах ГПП СПУ/ЕС

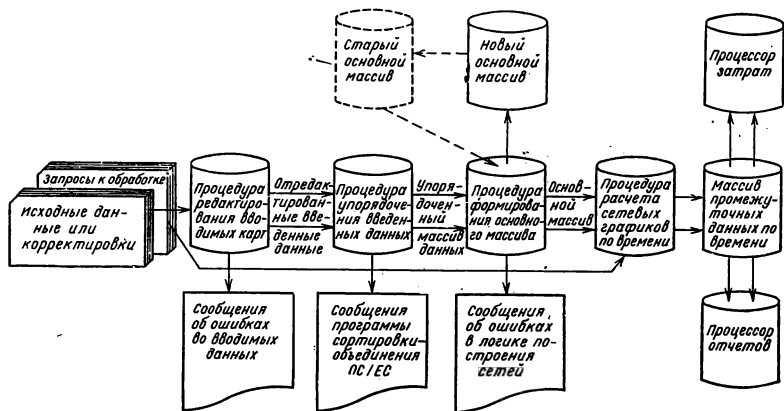


Рис. 31. Информационный поток в процедурах процессора сетей ППП СПУ/ЕС

В качестве примера приведем управляющую карту для вызова процедуры упорядочения вводимых данных *:

//STEP2 EXEC PGM= PORJADOK, COND=(4, LT, REDAKTOR)

Как видно из названий четырех перечисленных процедур, лишь последняя процедура выполняет функции непосредственной обработки данных. Три другие носят вспомогательный характер, освобождая тем самым пользователя от большого объема работы, предшествующей этапу непосредственного расчета по алгоритму поиска критического пути и т. п.

4. 3. 2. Структура процессора сетей

Процедура редактирования вводимых данных. Первый модуль, редактирующий входные данные, обеспечивает формальный контроль за соблюдением установленных правил описания вводимых данных, требуемых для обработки сети с помощью ППП СПУ/ЕС. Им обнаруживаются такие ошибки, как, например, наличие в сети более 250 подсетей, неправильные форматы вводимых данных и пр. При нахождении ошибок в исходных данных процедура редактирования действует в зависимости от запроса пользователя и характера самих ошибок. Некоторые ошибки могут быть отредактированы программой автоматически. При этом на печать выдаются ошибочные данные и их отредактированное содержание. Некоторые ошибки носят более серьезный характер, и их исправление должно быть проведено пользователем. Однако и в этом случае в зависимости от запроса пользователя процедура редактирования может обеспечить некоторые вспомогательные операции по ускорению процесса редактирования вводимых данных. Все ошибки подробно диагностируются. При этом пользователь получает текстовую информацию о характере ошибки, а также полный текст перфокарты, в которой обнаружена ошибка. Число различных типов диагностических сообщений, выдаваемых программой редактирования, — около 70. Результатом работы этой процедуры является отредактированный набор введенных данных.

Процедура упорядочения отредактированных данных. Поскольку ППП СПУ/ЕС обеспечивает обработку сетевых графиков, общее число работ в которых может достигать несколько десятков тысяч, при подготовке вводимых данных на перфокартах трудно упорядочить их вручную, как это требуется для нормальной работы программы. Поэтому функция упорядочения данных возлагается

* Каждая процедура ППП СПУ/ЕС по окончании работы вырабатывает код условия — число, характеризующее правильность выполнения программ: 0 — программа выполнена нормально; 4 — в исходных данных встречены небольшие ошибки; 8 — в исходных данных встречены серьезные ошибки, но программа выполнена и т. д. Условия, записываемые в параметре COND, определяют выполнение (или невыполнение) следующей процедуры (PORJADOK) в зависимости от кода условия предыдущей (REDAKTOR) [17]. В данном случае процедура упорядочения будет выполняться, если код условия, выработанный процедурой редактирования, меньше или равен 4.

на определенный модуль пакета — процедуру упорядочения, которая, решая эту задачу, подготавливает информацию для дальнейшей обработки. В процессе работы эта процедура использует сервисную программу сортировки-объединения ОС ЕС [17]. Результатом работы процедуры является отсортированный набор отредактированных данных.

Процедура формирования основного массива данных. Вводимые данные представляют собой в основном информацию о работах, которые необходимо провести для реализации конкретного проекта. Функцию формирования основного массива данных по сетевому графику выполняет третья процедура процессора сетей. В ходе ее выполнения анализируются входные данные и выявляются различные ошибки в логике построения сетевых графиков: «петли» в сетях, «висящие» начальные и конечные события и пр. Вся информация о выявленных ошибках представляется пользователю в виде диагностических сообщений, описанных выше. Общее число сообщений этой процедуры — около 30.

В дальнейшем данная процедура выполняет функцию коррекции созданного основного массива совместно с двумя предшествующими процедурами. В любом случае, если выполнение процедуры заканчивается успешно, создается новый или модифицируется имеющийся основной массив описания сетей.

Процедура обработки сетевых графиков. Четвертая процедура предназначена для обработки данных, хранящихся в основном массиве. Запрос пользователя указывает, какие сети и подсети и каким образом нужно обработать. В частности, запрос определяет: обрабатывать подсети как связанные или как обособленные, проводить ли детальную обработку или укрупнение сети и т. п.

Процедура заканчивается созданием промежуточного массива данных по временным характеристикам сети, который передается программам процессора отчетов для подготовки форм выходных документов или программам процессора затрат для последующей обработки.

Целью выполнения программ процессора сетей является создание (или модификация) основного массива, либо обработка основного массива по запросу пользователя, либо то и другое одновременно. В первом случае необходимо последовательное выполнение трех первых процедур процессора сетей. Результат их выполнения — новый основной массив, записанный на магнитную ленту, — может быть использован в дальнейшей работе. Во втором случае требуется реализация только последней процедуры. Для обеспечения такой возможности процедура обработки основного массива имеет собственные средства диагностики ошибок в логике построения сети.

С точки зрения машинной обработки процедуры процессора сетей полностью независимы. Поэтому при условии сохранения промежуточных массивов данных эти процедуры могут выполняться в разное время и даже на разных моделях ЕС ЭВМ, отвечающих требованиям ППП СПУ/ЕС и работающих с ОС ЕС. Однако с по-

зиций логики обработки сетевых графиков все процедуры в совокупности обеспечивают единый процесс, а каждая из них, взятая отдельно, не представляет целостной системы. Кроме того, связь процедур осуществляется путем передачи промежуточных массивов данных. При этом форматы данных, создаваемые передающими процедурами, легко воспринимаются принимающими модулями. В этом смысле процессоры пакета существенно отличаются от простых библиотек программ, где каждая программа может применяться обособленно и формирует обычно логически завершенные результаты обработки.

Подпрограммы — следующий уровень модульности пакета. Подпрограммы пакета реализуются как программные секции соответствующих модулей — процедур. В качестве примера задачи, решаемой с помощью подпрограммы, можно привести задачу генерации календаря с временным охватом, заданным пользователем.

Общее число подпрограмм сетевого процессора — более 50, поэтому их детальное описание нецелесообразно. Однако понятно, что достаточно мелкая разбивка алгоритма решения на модули

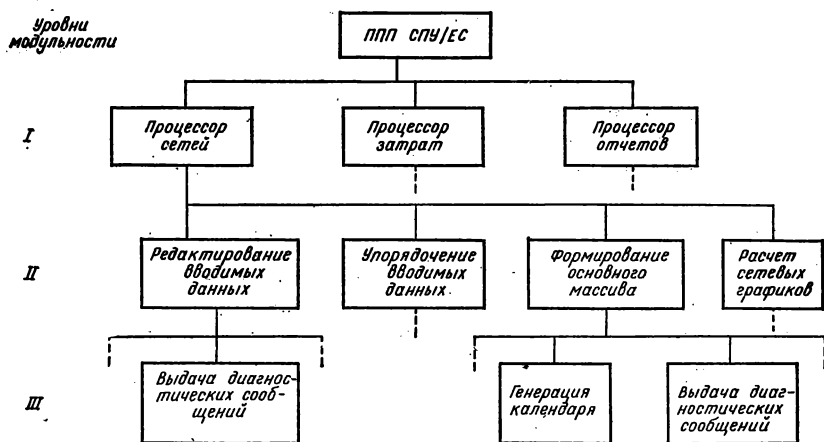


Рис. 32. Фрагмент модульной структуры ППП СПУ/ЕС:
I — процессоры; II — процедуры; III — подпрограммы

обеспечивает гибкость при обработке любого конкретного запроса пользователя.

При необходимости модификаций пакета пользователь может изменять и добавлять модули на любом из трех уровней: процессоры — процедуры — подпрограммы (рис. 32). При этом отдельные модули разработаны с учетом стандартных соглашений о связях программ, принятых в ЕС ЭВМ, и относительно независимы друг от друга, поэтому изменения в отдельных программах обычно не затрагивают остальной части программ пакета.

Так же как метод СПУ применим для реализации многих функций управления, так и подпрограммы, процедуры и процессоры

ППП СПУ/ЕС применимы в различных комбинациях. Конечно, как по техническим, так и по логическим соображениям нельзя собрать все программные модули на всех уровнях деления во всех теоретически возможных комбинациях. Тем не менее число различных вариантов работы программ, которые могут быть составлены по запросу пользователя из предоставленных типовых блоков, весьма велико.

4. 3. 3. Структура процессора отчетов

Основная функция процессора отчетов состоит в создании форм выходных документов на основе данных, получаемых в результате работы программ других процессоров пакета. Гибкость процессора отчетов в плане создания форм документов, соответствующих требованиям пользователя, можно отчасти охарактеризовать числом программ, входящих в него (их около 80).

Фазы создания выходной формы. Создание любой выходной формы с помощью программ процессора отчетов может быть разделено на две фазы: вызов отчета и генерация отчета.

Фаза вызова отчета обеспечивает выполнение таких задач, как прием промежуточных данных от других процессоров, подготовка памяти для них, контроль форматов данных и отбор данных, необходимых для создания выходных форм, затребованных пользователем.

В процессе генерации происходит непосредственно создание массива выходных форм: в соответствии с требованиями выходных форм сортируются данные в массивах, образованных в процессе вызова отчета, осуществляются необходимые алгебраические и логические операции над элементами данных в записях этого массива. После того как выходной массив организован, он выдается на печать.

Модульность процессора отчетов. Все программы процессора отчетов в соответствии с фазами формирования отчета можно разделить на программы стадии вызова и программы стадии генерации отчетов. С другой стороны, вся совокупность программ может быть естественным образом разделена на функционально-ориентированные и общего назначения.

Функционально-ориентированные программы специально предназначены для создания отчетов по данным, формируемым либо сетевым процессором, либо процессором затрат, либо при совместной обработке данных по времени и затратам, либо при независимых применениях*.

Программы общего назначения (программы для реализации арифметических операций, обеспечивающие выполнение соответст-

* Процессор отчетов представляет настолько мощное средство, что может рассматриваться как самостоятельная программная система, предназначенная для создания различных выходных форм. В отличие от языка РПГ [17] данная программа более проста в эксплуатации, хотя и менее универсальна.

вующих операций над любыми элементами данных произвольных записей *, программы ввода-вывода и др.) одинаково используются в любых применениях.

Таким образом, для процессора отчетов описан общий метод разбиения программы на модули (рис. 33), решающие отдельные задачи. Заметим, что подобный способ разбиения во многом обеспечивает ту гибкость прикладной программы, о которой говорилось выше.

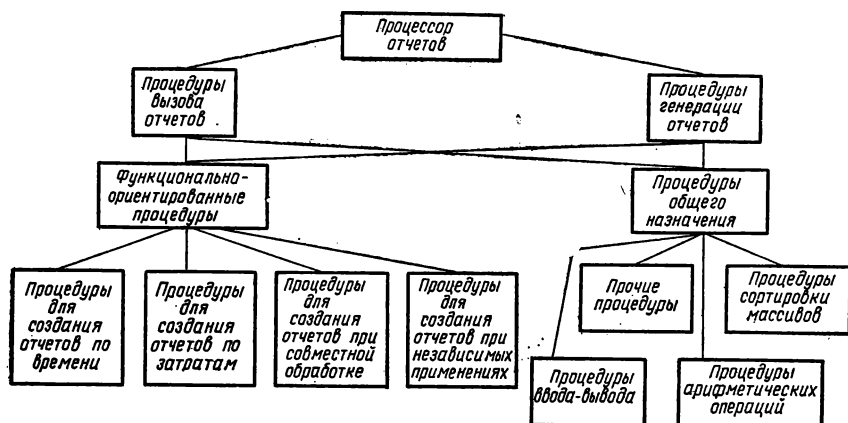


Рис. 33. Принцип разделения процессора отчетов на модули для обеспечения гибкости программы

4. 3. 4. Языковые средства для работы с ППП СПУ/ЕС

Для полного описания пакета прикладных программ СПУ/ЕС необходимо рассмотреть те языковые средства, которые имеет пользователь для эксплуатации пакета. При этом здесь дается лишь общее представление об этих средствах, поскольку более детальное их описание является основой для установки и эксплуатации программной системы и будет приведено в последующих параграфах данной главы.

При работе с программой используется набор языковых средств, который можно разбить на две группы. К первой относятся средства, принадлежащие самой прикладной программе, ко второй — язык управления заданиями операционной системы ОС ЕС.

Язык управления заданиями ОС ЕС как средство для работы с ППП СПУ/ЕС. Язык управления заданиями мы относим к средствам работы с программной системой по ряду причин.

Во-первых, операционная система представляет мощное средство, обеспечивающее реализацию многих функций, необходимых

* Такие процедуры необходимы на этапе генерации.

для работы программы. Применяя язык управления заданиями, пользователь описывает системе свои требования к выполнению программы, а также к входным, промежуточным и выходным массивам данных.

Таким образом, описание задания на языке управления можно рассматривать как управляющую программу пакета, обеспечивающую ту или иную обработку вводимых данных. При этом с помощью языка управления заданиями можно описать последовательность обработки данных не только на случай нормальной работы программы. С помощью параметра COND управляющего оператора EXEC могут быть описаны действия, которые необходимо выполнить операционной системе при появлении тех или иных ошибок в работе программы или во вводимых данных.

Другой широко употребляемый в ППП СПУ/ЕС метод, обеспечивающий гибкость программы путем использования средств языка управления заданиями операционной системы, — передача списка параметров из поля PARM оператора EXEC в выполняемые процедуры [17]. Таким образом могут быть определены некоторые задачи обработки данных. В качестве примера можно привести возможность определения таким способом объема и характера задач по редактированию вводимых данных при выполнении процедуры редактирования: необходимость распечатки введенных данных, редактирования небольших ошибок и т. п. Практически указанием параметров определяется набор и последовательность выполнения подпрограмм процедуры.

В процессе обработки данных с помощью ППП СПУ/ЕС широко используется программа сортировки-объединения, которая обеспечивает почти все функции по сортировке массивов данных, требуемые для работы пакета. Обращение к этой программе осуществляется внутрипрограммными средствами и средствами языка управления заданиями.

Таким образом, на языке управления заданиями реализуются многообразные связи программ пакета с операционной системой, которые во многом обеспечивают эффективность его работы.

Языковые средства ППП СПУ/ЕС. Языковые средства самой программы можно разделить на группы, соответствующие процессорам пакета. Практически каждый процессор имеет собственные языковые средства для своей работы. Эти средства используются для реализации двух основных функций: описания данных, подлежащих обработке программами соответствующего процессора, и описания требований к обработке этих данных (запросов к обработке). Обращение к пакету реализуется с применением языка управления заданиями и языковых средств, принадлежащих тем процессорам, которые используются при обработке данных. Принцип работы пакета и его взаимодействие с операционной системой показаны на рис. 34.

Помимо перечисленных средств ППП СПУ/ЕС имеет специальный макроязык, предназначенный для модификации стандартных форм выходных документов, обеспечиваемых программой. Этот

язык содержит гораздо большее число операторов и более сложен, чем те языковые средства, о которых говорилось выше.

Макроязык не предназначен для постоянного использования, а служит лишь для разработки специальных форм выходных документов, соответствующих требованиям пользователей. Однажды разработанные с использованием этого языка программы применяются для создания форм выходных документов постоянно, пока эти формы используются в данной системе управления.

Макроязык практически представляет собой специализированный язык, который существенно повышает гибкость системы. При

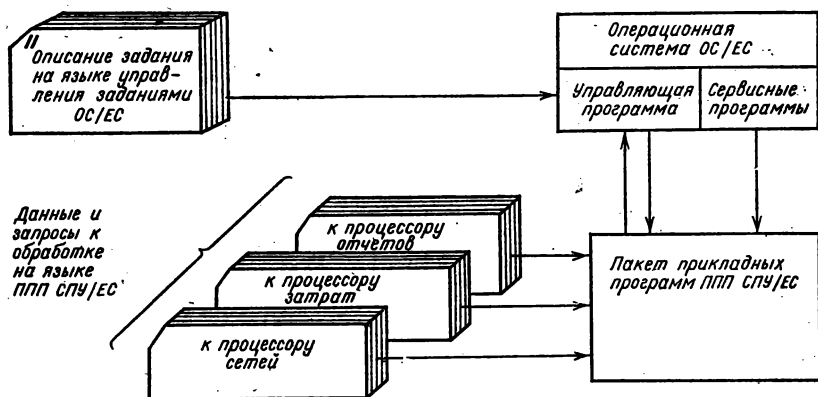


Рис. 34. Взаимодействие ППП СПУ/ЕС с операционной системой

его использовании трудоемкость привязки пакета к специфике задач конкретного пользователя существенно сокращается: модификация программы, разработанной другим программистом, почти всегда сложнее, чем разработка собственной небольшой программы на узкоспециализированном языке программирования. Кроме того, далеко не все формы выходных документов, которые могут понадобиться пользователю, можно получить простой модификацией имеющихся программ. Это обусловлено, с одной стороны, многообразием форм выходных документов, значительно превышающим многообразие методов обработки сетевых графиков, а с другой — сложностью самих программ процессора отчетов.

Таким образом, ППП СПУ/ЕС по архитектуре представляет собой типичную программную систему, снабженную специализированным языком для создания небольших программ, обеспечивающих формирование выходных документов. Управляющая программа этой системы реализуется средствами операционной системы.

От пакетов, имеющих структуру библиотеки программ, ППП СПУ/ЕС отличается последовательным использованием языковых средств в процессе эксплуатации программы. Кроме того, отдельные процедуры и подпрограммы пакета СПУ/ЕС не имеют самостоятельного значения для пользователя. Модули программы на

всех уровнях предназначены для решения определенных частей общей задачи и в этом плане существенно взаимосвязаны.

Данная система отличается по архитектуре от системы программирования с использованием специализированного языка более узкой сферой применения и тем, что языковые средства здесь в основном предназначены для описания данных и требований к обработке, а не алгоритма решения задачи*.

4. 4. ЯЗЫКОВЫЕ СРЕДСТВА ППП СПУ/ЕС КАК ОСНОВА ЕГО ЭКСПЛУАТАЦИИ

Языковые средства пакета являются основой для эксплуатации программы, и их освоение необходимо на этапе установки пакета.

Выше было сказано, что каждый процессор пакета обладает собственными языковыми средствами. Здесь будут описаны в основном языковые средства процессора сетей и лишь коротко — языковые средства процессора отчетов.

4. 4. 1. Языковые средства процессора сетей

Прежде чем описывать языковые средства, необходимо установить, какая информация о решаемой задаче требуется для работы программы. Естественно, что в большей части эта информация обусловлена самой экономико-математической моделью СПУ. Кроме того, могут учитываться специфические требования пользователя (например, желание ввести дополнительные элементы данных), а также особенности программной реализации. Ниже мы приводим основные типы данных, необходимых для работы программы.

Сети и подсети. Все сети и подсети, которые будут обрабатываться программой, должны иметь имена. При этом их имена в рамках одной сети не должны совпадать. Программе необходимо сообщить имя сети и имена всех подсетей, из которых она состоит. Кроме того, желательно указать дату последней модификации сети и подсетей.

Календарь. Для вычисления календарных дат, определяющих основные события разработки, необходимо задать информацию о годе начала разработки, общую продолжительность разработки, основную единицу времени, в которой будут выражены все оценки времени выполнения работ, дополнительные праздничные и рабочие дни на весь период разработки и на каждый год в отдельности.

Работы и события. Программа должна иметь всю необходимую информацию о работах, которые выполняются в процессе реализации проекта (имеются в виду работы, отмеченные стрелками в заданном сетевом графике). Информация о работах включает: имена предшествующих и последующих событий работ, а также

* Исключение составляет только макроязык для формирования программ процессора отчетов. Однако этот язык не применяется постоянно в процессе использования пакета, а требуется лишь на этапе его установки и модификации.

индексы работ в том случае, если между двумя событиями их количество превышает одну работу;

плановую дату наступления последующего события *, если она имеется, а также указание о необходимости вычисления вторичного резерва для предыдущих работ относительно этой даты;

информацию о том, является ли данная работа непрерывной или она может быть прервана на выходные и праздничные дни;

оценку времени выполнения работы (одну или три), выраженную в фиксированных единицах времени;

фактическую дату окончания работы, если она уже завершена;

шифр подразделения, ответственного за выполнение работы;

краткое словесное описание работы для облегчения чтения и использования выходных документов;

имя подсети, которой принадлежит данная работа.

Кроме того, должны быть описаны специальные события, т. е. начальные и конечные события сети, а также события, связывающие подсети. Информация о них включает:

имя события;

плановую дату наступления события, если она имеется;

указание о необходимости вычисления вторичного резерва от плановой даты;

фактическую дату наступления события;

краткое словесное описание события;

указание о том, является ли данное событие начальным, конечным или событием-связью;

имя подсети, которой принадлежит данное событие, и т. п.

Граничные события. Программа должна иметь полную информацию обо всех событиях сети, которые пользователь определил как граничные для дальнейшего укрупнения. Эта информация включает: уровень укрупнения, которому принадлежит данное событие, имя события, краткое словесное описание события, а также имя подсети, которой оно принадлежит.

Конец данных. При вводе данных необходимо указать программе на окончание колоды перфокарт с данными, описывающими конкретный сетевой график.

4. 4. 2. Задачи описания данных

Как видно из предыдущего, количество различных видов записей (типов данных), необходимых для описания сетевого графика, достаточно велико, не говоря уже о числе элементов данных, входящих в каждый тип. Поэтому при вводе исходной информации для работы программ пакета возникают две основные задачи: идентификация типа данных при первоначальном вводе данных с перфокарт, а также идентификация каждого элемента данных на перфокарте, описывающей данные определенного типа. При этом следует учитывать, что длины полей для некоторых элементов данных

* Если в событие, имеющее плановую дату, входит несколько работ, директивная дата может быть указана при описании одной из них (любой).

могут изменяться от одного применения программы к другому. Так, имя события может выражаться набором символов, содержащим от одной до девяти позиций в зависимости от способа кодирования событий в сетевом графике и объема самого графика.

Идентификация типов данных. Эта задача решается путем пробивки в первых двух позициях перфокарты идентификатора, соответствующего типу описываемых в перфокарте данных. Этот идентификатор представляет собой двузначное число и характеризует «тип карты», соответствующий типу описываемых в ней данных. Так, перфокарта, в первых колонках которой пробиты цифры 01 (карта типа 01), должна содержать информацию о сети и подсетях, а карта типа 05 — информацию для описания календаря и т. п. По двум первым позициям перфокарты программы процессора сетей распознают тип данных, вводимых на этой карте.

Идентификация элементов данных на картах различных типов. Эта задача решается следующим образом. Для каждого типа карт фиксируется последовательность представления элементов данных. Далее с помощью специальной карты типа 00 (карта-описатель) указываются начальные позиции и длины различных элементов данных в картах разных типов, обрабатываемых программами процессора сетей. Изменяя в определенных пределах данные в карте-описателе, пользователь может по своему усмотрению образовать форматы карт всех других типов. При этом формат самой карты-описателя фиксирован и не может изменяться. Это означает, что в определенных колонках карты-описателя содержится четко определенная информация. Например, в колонках 24 и 25 указывается длина поля словесного описания работы или события на картах работ и событий (тип 30), а в колонках 53 и 54 — начальная позиция этого элемента данных в картах типа 30 и т. д. (рис. 35).

После фиксации в карте-описателе форматы карт любых других типов не могут быть изменены, и какие-либо расхождения в оформлении этих карт с принятыми в карте-описателе правилами будут выявлены в процессе работы процедуры редактирования вводимых данных как ошибки.

Каждая логически самостоятельная единица данных (запись), определяемая набором элементов данных (например, описание определенной работы), требует отдельной карты. Таким образом, например, число карт типа 01 равно числу сетей и подсетей, обрабатываемых в рамках одного задания. Карта типа 00 всегда является единственной для всей сети.

При построении колоды перфокарт, описывающих вводимые данные, порядок следования карт различных типов фиксируется лишь для карты-описателя (первая в колоде), карт описания сети и подсетей — карт типа 01 (они следуют за картой-описателем), а также карты конца данных (последняя в колоде описания вводимых данных). Карты любых других типов могут следовать в колоде в произвольном порядке.

Описание требований к обработке. Языковые средства процессора сетей, предназначенные для описания запроса обработки,

Различные виды обработки сетей, реализуемые процедурой обработки процессора сетей, следующие:

- обработка сети в целом или отдельных подсетей;
- связанная или обособленная обработка подсетей в сети;
- детальная обработка или укрупнение сети и подсетей.

При укрупнении сети и подсетей возможны различные уровни укрупнения.



Все эти данные должны найти отражение в карте запроса обработки. Число таких карт при одном выполнении программы не фиксируется, так что одно задание обеспечивает различные виды обработки для различных подсетей в сеге.

Таковы основные принципы построения языковых средств процессора сетей, обеспечивающие эксплуатацию программы.

4. 4. 3. Языковые средства процессора отчетов

Эти средства предназначены только для запроса необходимой обработки. При этом сама обработка данных программами процессора отчетов состоит в формировании выходных документов на основе результатов работы программ предыдущих процессоров. Языковые средства процессора отчетов определяются несколькими различными типами карт. Ниже рассмотрен лишь один наиболее распространенный тип карт — карта запроса отчетов.

Программы процессора отчетов создают различные стандартные формы выходных документов. При этом общее число таких форм, различных с точки зрения компоновки данных и использования этих форм — десять (по системе СПУ-ВРЕМЯ). Однако в зависимости от требований запроса обработки любая из этих форм может быть приспособлена к описанию детальных или укрупненных сетевых графиков. Таким образом, общее число различных подготавливаемых отчетов достаточно велико и содержание их достаточно разнообразно для того, чтобы обеспечить пользователя необходимой информацией по управлению.

Сведения о номере требуемого отчета, уровне укрупнения сети или подсетей, а также имена сети и подсетей, по которым должен быть составлен отчет, представляют основную информацию, помещаемую пользователем в карту запроса обработки. Данная карта имеет фиксированный формат и предназначена исключительно для запроса определенных форм отчетов. Общее число таких карт в одном задании не фиксируется, так же как и общее число карт запроса обработки в сетевом процессоре.

4. 4. 4. Сравнение языковых средств панета с алгоритмическими языками программирования

Приведенные здесь средства и методы работы с пакетом существенно отличаются от общеизвестных языков программирования не только своей простотой, которая обусловлена относительной простотой алгоритмов решаемой задачи, но и принципами использования. Рассмотрим более подробно, чем обусловлены эти отличия, поскольку это существенно не только для понимания различий между программными системами и системами программирования с использованием специализированных языков, но и для понимания

принципов функционирования и эксплуатации программ, имеющих архитектуру программной системы.

Первое важное отличие проявляется в том, что языковые средства рассматриваемого пакета предназначены для описания вводимых данных и запросов к обработке, а не для описания алгоритмов вычислений. Сами эти алгоритмы в такой степени детерминированы, что определяются однозначно уже из описания задания на языке управления заданиями, а также из требований пользователя к обработке, представленных средствами языка запроса.

Специализированные языки программирования позволяют описать достаточно сложные виды обработки данных при использовании отдельных операторов языка. Так, некоторые языки обеспечивают обычные алгебраические операции над матрицами путем применения одного оператора языка для каждой операции. Однако, как правило, программа на специализированном языке не предназначена для выполнения каких-либо операций, реализуемых одним-двумя операторами языка, а обеспечивает решение задачи, в которой отдельные достаточно сложные по выполняемым действиям операторы языка реализуют лишь некоторые части общего алгоритма решения.

Простота языковых средств ППП СПУ/ЕС и ограниченность его возможностей даже при реализации функций описания вводимых данных и запросов обработки обусловлены относительной определенностью алгоритмов самой решаемой задачи, а также относительно малым объемом и простотой структур используемых данных. Однако для ППП СПУ/ЕС описанные средства вполне достаточны при решении задач управления. С другой стороны, простота этих средств является преимуществом программы, поскольку она предназначена для массового применения.

Чтобы подчеркнуть простоту используемых средств, мы почти нигде не употребляли термин «язык», а пользовались общим понятием «языковые средства». Практически это набор средств, позволяющих эксплуатировать прикладные программы широкому пользователю, имеющему лишь самые общие представления о программировании и ЭВМ в целом. С усложнением задач, решаемых программной системой, усложнятся и используемые языковые средства. Так, более сложными являются такие средства для работы с банком данных, который используется программистом при создании интегрированных систем обработки данных. Это объясняется более сложными структурами данных в таких системах, а также сложностью самих возможных запросов, которые система управления базой данных должна удовлетворять. Однако подобные языковые средства работы с банком данных, несмотря на большую сложность, выполняют те же самые функции, что и языковые средства рассматриваемого пакета: описание данных и запросов к обработке. При этом следует иметь в виду, что понятие обработки здесь не означает непосредственную обработку данных пользователя (система управления базой данных для интегрированной СОД не ведет такой обработки), а служит для определения действий,

выполняемых системой по запросу прикладной программы *, таких, как поиск и выдача необходимых данных, реорганизация структуры базы данных и т. п.

Таким образом, описание языковых средств пакета прикладных программ СПУ/ЕС можно рассматривать как типичный, в определенной мере, пример языковых средств прикладной программы среднего уровня сложности, имеющей структуру программной системы.

4. 5. ВОПРОСЫ УСТАНОВКИ И ПРИМЕНЕНИЯ ППП СПУ/ЕС

На этапе установки пакета в систему обработки данных, который всегда предшествует этапу его эксплуатации, должны быть решены вопросы привязки пакета к условиям его реализации у конкретного пользователя, а также вопросы организации всей исходной информации, которая в дальнейшем служит базой для эксплуатации программы. При этом привязка пакета осуществляется по двум основным направлениям: привязка к той конфигурации вычислительных средств, которая имеется у пользователя, и привязка к специфике реализации задачи у пользователя.

4. 5. 1. Привязка программы к конфигурации вычислительных средств

Привязка программы к существующей конфигурации вычислительных средств проводится с целью обеспечения эффективного функционирования программы на данной вычислительной установке. При этом эффективность можно определить как с позиций функционирования системы обработки данных и вычислительной системы в целом, так и с позиции удовлетворения требований пользователя к решению задач сетевого планирования. Эти два подхода к определению эффективности существенно взаимосвязаны. Так, скорость решения задач с использованием ППП СПУ/ЕС во многом обуславливается объемом оперативной памяти, выделяемой для выполнения программы, однако эффективное функционирование всей системы обработки данных в целом требует снижения объемов оперативной памяти, предназначенной для решения отдельных задач пользователей. Общая эффективность и в соответствии с этим конкретная реализация программы в системе обработки данных достигаются при учете таких факторов, как объем, сложность и важность задач СПУ в ряду тех задач, которые пользователь решает с применением вычислительной машины. При этом сам пакет предоставляет в распоряжение пользователя набор средств, обеспечивающих необходимую гибкость программ с позиций их привязки к существующей вычислительной системе с наибольшей эффектив-

* Заметим, что языковые средства банка данных, предназначенные для непосредственной работы с ним пользователя, обычно более просты.

ностью. Здесь мы исходим из того, что эффективность выполнения программы, определяемая скоростью ее работы, объемами и числом обрабатываемых сетей и подсетей, количеством лет, охватываемых календарем, обеспечением необходимых размеров некоторых важных элементов данных (таких, как длина поля имени события и пр.), является функцией размеров ресурсов вычислительной системы, выделенных для ее выполнения. Для ППП СПУ/ЕС основными ресурсами являются объемы оперативной и внешней памяти. Поэтому далее мы будем говорить об эффективном использовании именно этих средств.

Реализация эффективной работы программы. Документация пакета включает ряд формул, связывающих основные параметры работы программы с необходимыми объемами областей основной и внешней памяти. Используя эти формулы, а также учитывая требования конкретной системы управления разработкой к объему и структуре детального сетевого графика, программисты-пользователи и системные программисты, занятые установкой пакета, могут определить, с одной стороны, оптимальный объем требуемой памяти, а с другой — оптимальные значения параметров, определяющих вариант работы программы. При этом работа программы в различных по объему областях оперативной памяти обеспечивается возможностью организации как простой структуры, так и структуры с запланированным перекрытием [17] для некоторых процедур пакета. Так, если организуется простая структура для процедур процессора сетей, требуется 25К для процедуры редактирования вводимых данных, 3,5К (и дополнительно память для программы сортировки-объединения ОС ЕС) для процедуры упорядочения данных, 37К для процедуры формирования основного массива и 35К для процедуры обработки сетевых графиков. Здесь объемы памяти приведены без учета требований обрабатываемых массивов данных, которые рассчитываются дополнительно в зависимости от размеров сетей и подсетей.

Область памяти, необходимая для программ процессора сетей, определяется по области самой большой процедуры — 37К, так как все процедуры выполняются последовательно в рамках одного задания. Две последние, наибольшие по требуемому размеру основной памяти процедуры процессора сетей, могут быть организованы как с использованием простой структуры, так и с использованием структуры с запланированным перекрытием*, обеспечивающей вызов в память по мере необходимости отдельных подпрограмм соответствующих процедур [17]. В этом случае объем памяти, требуемой для процедур, сокращается до 25К или даже до 16К в зависимости от структуры перекрытия. Программы, имеющие структуру с запланированным перекрытием, требуют для выполнения больше времени, однако при этом снижаются требования к объему опера-

* В данном случае термин «структура программы» отражает принципы ее функционирования с позиций операционной системы и не связан прямо с понятием «архитектура», введенным выше. Подробно о возможных структурах программ в этом значении данного понятия см. [17].

тивной памяти, которая в определенных случаях может оказаться в системе обработки данных более важным ресурсом, чем время работы процессора. Документ «Руководство программиста» содержит инструкции, описывающие процесс организации процедур со структурой запланированного перекрытия.

Требования к объему внешней памяти на устройствах прямого доступа, а также косвенно требования к объему оперативной памяти указываются в описании задания на языке управления заданиями [17]. Кроме того, требования к оперативной и внешней памяти сообщаются оператору вычислительного центра с тем, чтобы на их основе он определил раздел основной памяти, в котором данное задание будет выполняться, а также установил состав внешних устройств, участвующих в его выполнении.

Изменения в обрабатываемых сетевых графиках или в конфигурации вычислительных средств могут вызвать необходимость изменения размеров ресурсов, выделяемых программам пакета для обработки данных. Эти изменения ресурсов призваны обеспечить наибольшую эффективность решения задач пользователя в новых условиях.

4. 5. 2. Привязка программы к специфике задач пользователя

Другим важным вопросом этапа установки ППП СПУ/ЕС является привязка программ пакета к специфике требований конкретного пользователя при решении задач сетевого планирования и управления. Поскольку эта специфика не оказывает обычно существенного влияния на экономико-математическую модель СПУ, т. е. на методы расчета критического пути, резервов времени и т. п., а в основном связана с различными режимами проведения работ (числом рабочих дней в неделю, числом рабочих часов в день, дискретным или непрерывным характером производства и т. п.), программы пакета без специальных модификаций привязываются к требованиям пользователя. Привязка в основном осуществляется путем указания необходимых параметров работы программы, а также форматов для карт различных типов. Эти форматы фиксируются в карте-описателе. В дальнейшем при работе с сетевым графиком пользователь уже не может изменить эти форматы без реорганизации основного массива. Поэтому при установке пакета к проектированию форматов данных необходимо внимательно подойти с самого начала.

Часть ППП СПУ/ЕС, где изменения наиболее вероятны, — создание форм выходных документов. Здесь модификации могут осуществляться, например, для установления соответствия между существующими в данной системе управления формами документов и формами, которые создаются программами пакета в процессе их работы. Таким образом, в большинстве случаев изменения могут касаться программ, принадлежащих процессору отчетов пакета

СПУ/ЕС. Для упрощения подобных модификаций пакет включает макроязык, средствами которого пользователь может описать требуемые ему формы выходных документов.

Макроязык процессора отчетов представляет собой набор специально разработанных макрокоманд языка Ассемблера ОС ЕС [17]. Каждая макрокоманда выполняет определенную функцию по составлению выходной формы, например отбор данных для отчета по указанным пользователем критериям, сортировку данных в массиве отобранных данных по указанным элементам данных и др.

При подготовке новых выходных форм программисты, занятые установкой пакета, совместно с пользователем детально разрабатывают эти формы. При этом для всех элементов данных, которые будут отражены в отчете, должны быть указаны начальные и конечные позиции на формате листа, получаемого с устройства печати. Следует учесть также расположение заголовков формы, пропуски строк, если это необходимо, и др. Разработанная форма принимается программистом, который описывает порядок ее реализации на макроязыке процессора отчетов. При этом, как и в случае подготовки стандартных отчетов, создаются две программы: вызова и генерации. Описание макроязыка, содержащееся в документе «Руководство программиста», обеспечивает всеми необходимыми сведениями по составлению этих программ.

Разработанные программы транслируются, а затем обрабатываются редактором связей ОС ЕС [17]. При этом, поскольку в разработанной на языке Ассемблера программе встречаются несистемные макрокоманды [17], в описании задания на трансляцию на языке управления заданиями нужно указать информацию о библиотеке макроопределений для соответствующих макрокоманд. Эта библиотека макроопределений для макрокоманд процессора отчетов наряду с другими процессорами является частью ППП СПУ/ЕС. В отличие от остальных библиотек (процессоров) пакета, содержащих выполнимые (загрузочные) модули [17], данная библиотека содержит модули, написанные на языке Ассемблера. Она используется только в случае составления новых программ создания форм отчетов и не требуется в процессе обработки данных пользователя.

После трансляции и обработки редактором связей новая программа создания отчета может быть записана под каким-либо именем в процессор отчетов. При обработке данных по сетевым графикам пользователь может вызвать ее с помощью карт запроса отчетов, указав имя соответствующей программы. Программа подключается к работе и формирует необходимый отчет.

Таким образом, на этапе установки, используя средства макроязыка, можно легко модифицировать те программы пакета, которые чаще всего нуждаются в изменениях.

Описание исходных данных. Заключительный этап работ по установке пакета включает описание всей информации по сетевому графику с использованием языковых средств процессора сетей,

исправление ошибок в подготовке исходных данных и формирование основного массива.

Первая часть работы состоит в кодировании информации о сетевом графике. Здесь необходимо проанализировать исходную информацию по типам данных с целью определения форматов карт всех типов. Эти форматы могут варьироваться в определенных пределах от одного применения к другому. Однако в каждом конкретном применении они должны быть фиксированы для карт всех необходимых типов. Например, все карты, содержащие информацию о работах сетевого графика (карта типа 30), должны иметь одинаковый фиксированный формат.

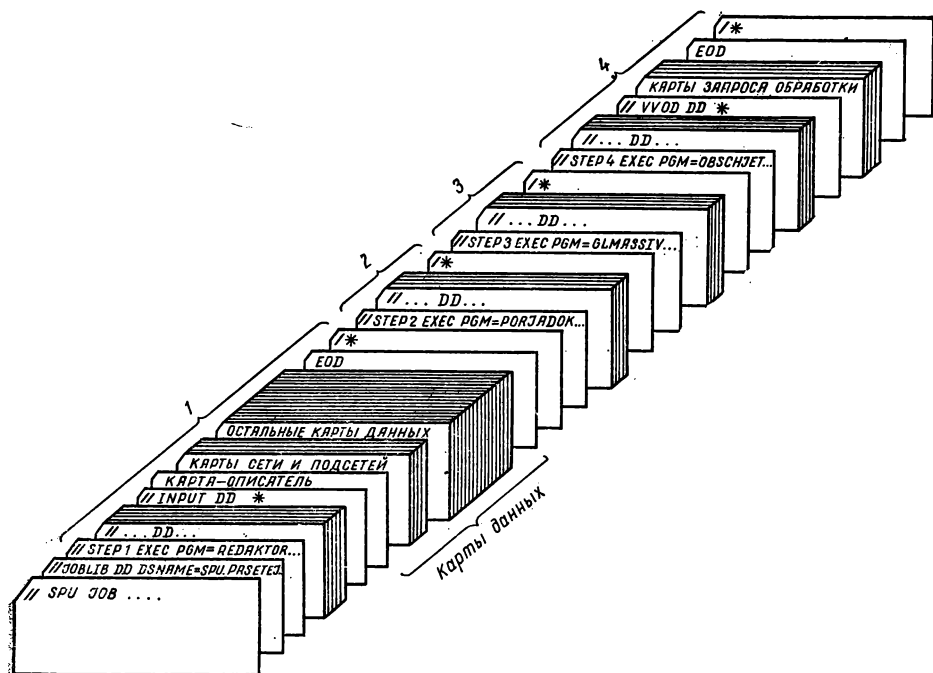


Рис. 36. Порядок карт в задании для процессора сетей (без каталогизации):
1 — карты процедуры редактирования; 2 — карты процедуры упорядочения; 3 — карты процедуры создания главного массива; 4 — карты процедуры обработки

После установления форматов полученные для них значения должны быть занесены в карту-описатель, которая окончательно закрепляет их на все время работы с основным массивом данных. В соответствии с принятыми ограничениями заполняются карты остальных типов. При этом особое внимание следует уделять описанию граничных событий для последующего укрупнения сетей. Эти события с привязкой к уровням укрупнения сетей должны быть определены заранее в процессе разработки сетевого графика с учетом того, чтобы наилучшим образом отразить структуру сети после укрупнения.

Вся исходная информация о сети и описание задания на языке управления заданиями после перфорации собираются в виде колоды перфокарт. На рис. 36 представлена колода, включающая карты языка управления заданиями для описания процессора сетей и четырех его процедур. Карты с описанием каждой процедуры полностью независимы, поэтому основной массив может создаваться при задании полного набора карт, исключая карты процедуры обработки, а обработка (при условии наличия основного массива) может вызываться картами языка управления заданиями, описывающими процессор сетей и процедуру обработки, исключая карты с описанием остальных процедур.

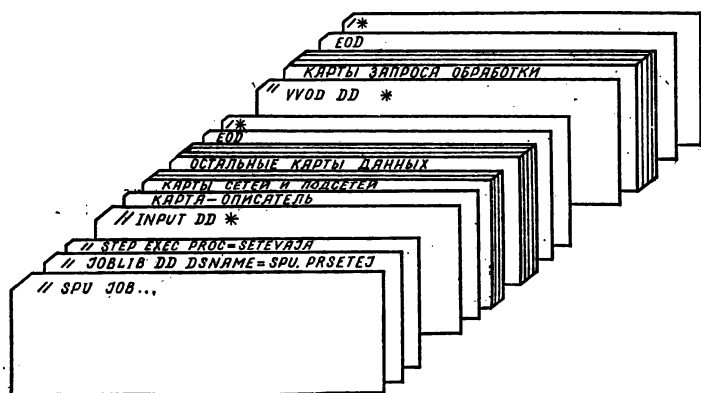


Рис. 37. Порядок карт в задании для процессора сетей (с каталогизацией)

При создании основного массива сети возможны ошибки в исходных данных и логике построения сетевого графика. При наличии таких ошибок программы не создают основной массив, а с помощью диагностических сообщений информируют пользователя, в каких именно картах встретились ошибки и каков их характер. Выявление и устранение всех ошибок, как правило, реализуется в два-три этапа. Каждый этап состоит в выполнении программ процессора сетей, анализе полученных диагностических сообщений и корректировке исходных данных.

Для облегчения и упрощения работы с программой ее можно каталогизировать, т. е. записать всю основную управляющую информацию о задании на резидентный том вычислительной системы [17]. Это позволит существенно уменьшить объем колоды перфокарт, требуемых для каждого выполнения программы (рис. 37). При необходимости процедура обработки данных может быть каталогизирована отдельно.

Некоторые вопросы применения ППП СПУ/ЕС. Важнейшим результатом использования ППП СПУ/ЕС в процессе управления разработкой является возможность получения отчетов о состоянии разработки, обеспечивающих руководителей необходимой информа-

цией. В процессе эксплуатации пакета может быть установлена определенная периодичность формирования выходных документов, которая может учитываться при планировании общей загрузки вычислительной системы.

Формирование отчетов на основе массива, создаваемого процедурой обработки сетевых графиков, может осуществляться либо в рамках отдельного задания, либо в одном задании с организацией или модификацией основного массива. Во втором случае з описании задания на языке управления заданиями вслед за картой //JOBLIB DD... (см. рис. 36, 37), описывающей процессор сетей как личную библиотеку пользователя, должна идти карта // DD..., аналогичным образом описывающая процессор отчетов [17].

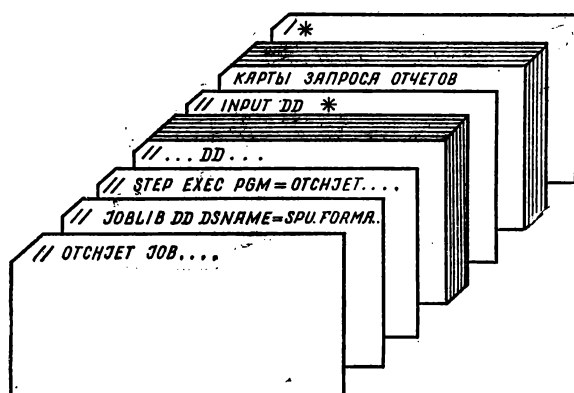


Рис. 38. Порядок карт в задании для процессора отчетов

Последовательность карт в отдельном задании, предназначенном для формирования стандартных отчетов, показана на рис. 38. В картах запроса указываются номера форм документов, необходимых пользователю. Описание стандартных форм документов по системе СПУ-ВРЕМЯ дано в приложении 2.

Спецификой задач сетевого планирования и управления является итеративный метод разработки сетевого графика, т. е. последовательное перепланирование с целью выбора наиболее приемлемого варианта сети [57]. Кроме того, в дальнейшем в процессе управления требуются регулярный контроль и корректировка сетевого графика, связанные с модификациями основного массива, созданного процедурами сетевого процессора. Практически эта работа, а также подготовка отчетов о состоянии работ по сетевому графику являются важнейшими при эксплуатации ППП СПУ/ЕС.

Пакет СПУ/ЕС предусматривает выборочную коррекцию любых необходимых фрагментов сетей и подсетей с целью создания окончательного сетевого графика, на базе которого формируется оперативно-календарный план разработки. Одновременно с формирова-

нием графика создается основной массив данных о работах, который программы пакета используют в дальнейшем для обработки.

При модификации основного массива подготавливаются данные о тех элементах сети и подсетей, которые должны быть изменены. Эти данные заносятся на карты соответствующего типа в форматах, определенных в карте-описателе при создании основного массива. Полученная колода перфокарт формируется в порядке, указанном на рис. 36 или 37. При этом наряду с картами, в которых имеются изменения, в колоде обязательно должны присутствовать карты типа 00 и 01.

4. 5. 3. Совершенствование программной системы

В процессе эксплуатации прикладной программы должно уделяться постоянное внимание контролю процесса эксплуатации и выявлению новых требований системы управления к программной реализации. Эта работа касается как организационно-технической, так и содержательной части задач. В плане содержательной части задачи необходим целенаправленный и систематический анализ функционирования самой системы управления разработкой. При таком анализе обеспечивается динамическое соответствие требований, предъявляемых к системе управления в каждом конкретном случае, с одной стороны, и степени их удовлетворения конкретной программой — с другой. Результатом этой работы может явиться изменение каких-либо аспектов системы управления, которое, возможно, найдет свое отражение в модификациях программ пакета.

Другим аспектом данной проблемы являются технические вопросы оценки работы программы в связи с изменениями в требованиях к решению задач, а также изменениями в конфигурации вычислительных средств. При этом подобная регулярная оценка необходима для повышения эффективности обработки данных с помощью пакета. В частности, важнейшей характеристикой эффективности эксплуатации программы является время ее работы на некотором массиве данных определенного стандартного объема. Такие характеристики позволяют, с одной стороны, оценить возможности программы в плане обработки данных и, с другой, определить эффективность ее эксплуатации в условиях конкретного применения. При этом следует учесть, что подобные оценки могут быть приближенными и существенно зависят от конфигурации вычислительной системы и характера данных, с которыми ведется работа. Последнее в случае обработки сетевых графиков в какой-то мере определяется количеством работ и путей в сети, а также их взаимосвязями.

В целом можно привести характеристики для каждой процедуры сетевого процессора и работы программ процессора отчетов по подготовке одной выходной формы (табл. 5).

При этом предполагается наличие вычислительной системы, содержащей два накопителя на сменных магнитных дисках (помимо

накопителей для резидентного тома, содержащего программы операционной системы), два накопителя на магнитных лентах, одно устройство ввода с перфокарт и одно печатающее устройство.

4. 5. 4. Документация ППП СПУ/ЕС

Важным вопросом, определяющим эффективность эксплуатации программы, является принцип построения системы документации, которая должна включать все необходимые сведения и характеристики для использования программы. Модульность построения про-

Таблица 5

**ЗАВИСИМОСТЬ ПРОИЗВОДИТЕЛЬНОСТИ ППП СПУ/ЕС
ОТ МОДЕЛИ ЕС ЭВМ (ВРЕМЯ РАБОТЫ ПРОГРАММ ПРИВЕДЕНО В МИН)**

	Модели и емкость оперативной памяти, К байт				Примечание
	ЕС-1020 128	ЕС-1030 256	ЕС-1040 256	ЕС-1050 512	
Процедуры сетевого процессо- ра:					
редактирование данных	1,5	1,5	1,5	1,5	1 000 карт
упорядочение данных	0,8	0,6	0,5	0,4	1 000 карт
формирование массива*	1,7	1,0	0,8	0,6	1 000 работ
	<u>2,3</u>	<u>2,0</u>	<u>1,0</u>	<u>0,7</u>	
обработка*	<u>4,0</u>	<u>3,0</u>	<u>2,0</u>	<u>1,2</u>	1 000 работ
	6,0	4,0	3,0	1,8	
Процессор отчетов	1,8	1,1	0,8	0,7	1 000 работ в одном отчете

* В числителе приведены данные для расчетов по одной подсети; в знаменателе — по семи подсетям.

граммы нашла отражение в структуре системы документации. Она позволила обеспечить независимость отдельных частей документации и удобство при выборе пользователем требуемых в работе возможностей программы (рис. 39).

Для каждой категории специалистов, занятых работой с ППП СПУ/ЕС, имеется документация. При этом выделяются группы специалистов в зависимости от их функций при работе с пакетом.

Специалисты различных уровней руководства разработкой и руководители вычислительного центра. Эти специалисты должны ознакомиться с возможностями программы и требованиями ее к конфигурации вычислительной системы. На данном этапе работ принимается решение об установке и эксплуатации программы или о ее модификации с целью учета определенной специфики задач пользователя, которая не обеспечивается стандартной версией программы.

Сведения, необходимые для решения этих вопросов, содержатся в вводном руководстве по ППП СПУ/ЕС — «Описание применения».

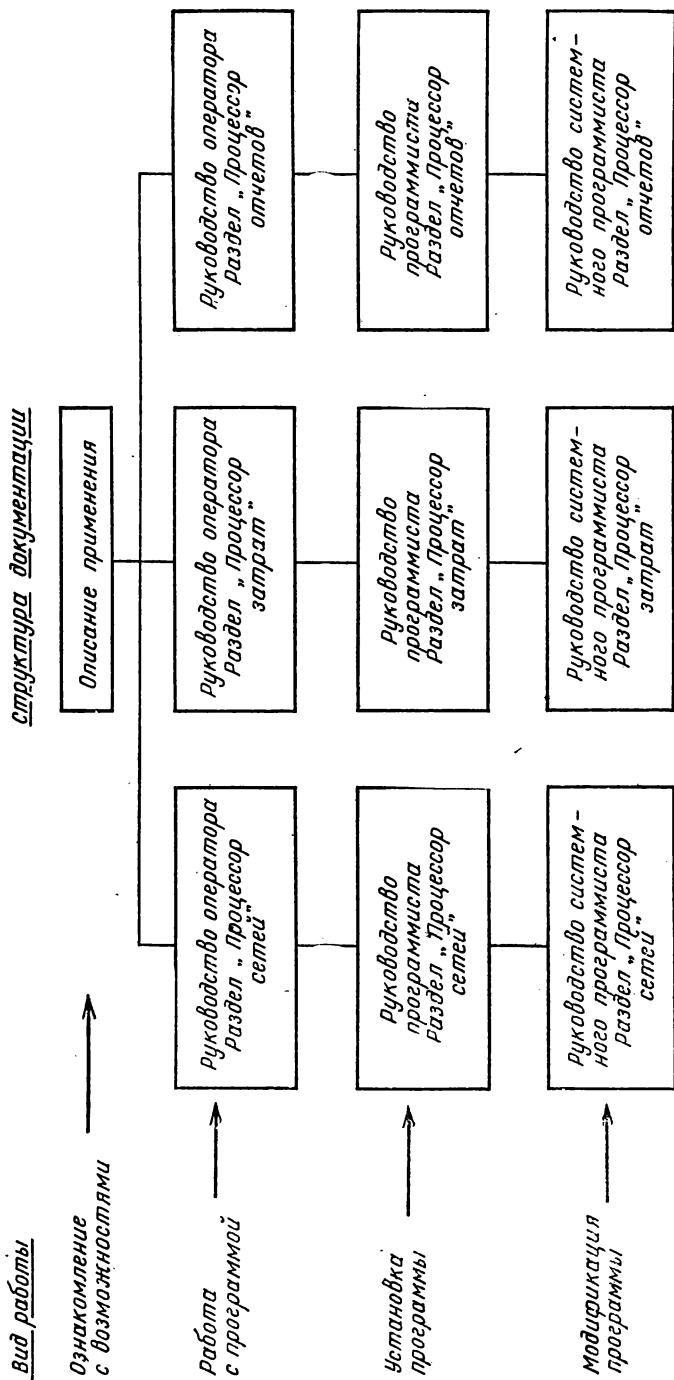


Рис. 39. Система документации ППП СПУ/ЕС

Персонал, обеспечивающий функционирование системы СПУ. Задачей этой категории специалистов является информационное обеспечение работы программы, т. е. своевременный сбор информации о состоянии разработки в соответствии с сетевым графиком и коррекция основных массивов, а также регулярное выполнение программы с целью получения выходных отчетов.

Все сведения, необходимые для выполнения этой работы, имеются в документе по эксплуатации ППП СПУ/ЕС — «Руководство оператора». Данный документ разбит на разделы таким образом, что сведения об эксплуатации каждого процессора могут изучаться независимо от материалов по другим процессорам. Это удобно в случаях, когда конкретные задачи сетевого планирования подразумевают независимое использование процессора сетей и процессора затрат.

Системные программисты и программисты-пользователи. В функции этой категории специалистов входят установка и модификация программ пакета. Вся информация, необходимая для установки, изложена в «Руководстве программиста», которое также разделено на три части — по числу процессоров пакета.

В документации, необходимой для проведения модификации программ пакета («Руководство системного программиста»), содержится описание принципов функционирования пакета на уровне подпрограмм. Системное руководство также разделено на три независимые части. В каждом документе подпрограммы собраны в группы, относящиеся к конкретной процедуре. В документе подробно описываются функции каждой подпрограммы, алгоритм, который она реализует, и особенности программирования. Данное руководство обеспечивает возможность простой модификации программ пакета на любом уровне модульности, поскольку позволяет квалифицированным специалистам быстро освоить основные идеи и методы построения пакета, заложенные разработчиками. Благодаря этому можно модифицировать программы пакета децентрализованно силами специалистов, не участвовавших непосредственно в разработке пакета.

Глава 5

ПРИКЛАДНЫЕ ПРОГРАММЫ И РЕЖИМЫ ЭКСПЛУАТАЦИИ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Режимы эксплуатации ЭВМ. В предыдущих главах прикладные программы рассматривались в основном с позиций требований пользователей и в связи с организацией интегрированных систем обработки данных. Целый ряд новых вопросов разработки и использования прикладных программ выявляется в процессе анализа различных режимов эксплуатации вычислительных машин. При этом под режимом эксплуатации ЭВМ здесь понимается принцип организации взаимодействия пользователя с машиной, который определяется возможностью влияния пользователя на процесс выполнения программ, а также числом пользователей, имеющих одновременный доступ к вычислительным средствам.

Подход к анализу прикладных программ с позиций режимов эксплуатации приобретает особое значение для ЕС ЭВМ, поскольку это семейство ЭВМ содержит набор средств, обеспечивающих широкое разнообразие режимов эксплуатации вычислительных машин. Так, в частности, ЕС ЭВМ включает ряд средств для организации режимов разделения времени [4], [17], [19]. Режимы разделения времени открывают новые возможности применения ЭВМ и в ближайшее время должны получить широкое распространение при использовании вычислительных машин в народном хозяйстве.

Ниже рассматриваются некоторые особенности прикладных программ, предназначенных для эксплуатации ЭВМ в режимах разделения времени. Отметим, что в отличие от предыдущих глав, где определенные сведения о вычислительных машинах предполагались известными читателю из других книг, в этой главе излагаются некоторые сведения об эксплуатации ЭВМ Единой системы в указанных режимах. Это вызвано тем, что режимы разделения времени не получили пока еще широкого распространения в СОД

пользователей. Кроме того, в условиях, когда терминология в данной области еще окончательно не установилась, целесообразно определить используемые в тексте термины и понятия. При этом отметим, что наряду с технической терминологией мы пользовались некоторыми понятиями, полезными в методическом отношении. В частности, это относится к понятиям «режим работы» и «режим эксплуатации» ЭВМ.

5. 1. РЕЖИМЫ РАБОТЫ И РЕЖИМЫ ЭКСПЛУАТАЦИИ ЭВМ

Важнейшими технико-экономическими характеристиками любой системы обработки данных являются производительность вычислительной системы и производительность труда программиста.

Производительность вычислительной системы обычно рассматривается не как синоним быстродействия процессора, а как некоторый обобщенный критерий, характеризующий эффективность работы машины. В отличие от быстродействия критерий производительности отражает не только технические характеристики центральной части ЭВМ (процессора, оперативной памяти и каналов ввода-вывода), но и характеристики внешних устройств, операционных систем и даже эффективность той или иной конфигурации вычислительной системы при решении различных задач.

Производительность вычислительных систем во многом зависит от технических принципов их реализации, которые мы описываем термином «режим работы» ЭВМ.

Под режимом работы ЭВМ мы понимаем те принципы структурной организации и функционирования технических средств и операционной системы ЭВМ, которые определяют ее производительность.

Производительность труда программиста также обусловлена целым рядом факторов. В частности, она зависит от квалификации программиста, сложности программируемой задачи и т. п. Поскольку отладка программы составляет значительную долю общего времени ее разработки, производительность труда программиста во многом определяется режимом эксплуатации ЭВМ, принятым в системе обработки данных. Режим эксплуатации в данном случае определяет принцип взаимодействия программиста с ЭВМ в процессе отладки программы.

В любой системе обработки данных факторы производительности ЭВМ и производительности труда программиста существенно взаимосвязаны. Эта связь обуславливает определенные соотношения между режимами работы и режимами эксплуатации ЭВМ. Рассмотрим подробнее возникающие здесь вопросы.

5. 1. 1. Производительность вычислительной машины

Количественное выражение показателя - производительности ЭВМ вызывает определенные затруднения. Это обусловлено большим числом факторов, влияющих на общую производительность

вычислительной системы, и их сложной взаимосвязью. При оценке производительности той или иной системы обычно проводится качественный анализ влияния различных факторов на общую производительность. Например, нетрудно заметить, что такая характеристика, как объем оперативной памяти, не влияет непосредственно на быстродействие ЭВМ. Однако она в значительной степени связана с производительностью машины. Действительно, если программа достаточно велика и не может быть размещена в оперативной памяти целиком, то единственный способ решения задачи состоит в разделении программы на отдельные блоки и выполнении их в определенной последовательности. При этом время решения задачи может значительно увеличиться по сравнению с ситуацией, когда программа целиком размещается в памяти, за счет дополнительного времени, необходимого для организации связи блоков и их ввода в ЭВМ с внешних носителей. Таким образом, с увеличением объема оперативной памяти ЭВМ ее производительность возрастает (до определенного уровня). В дальнейшем мы будем возвращаться к анализу работы вычислительных машин с позиций производительности, поскольку она является важным критерием оценки эффективности работы ЭВМ и позволяет выявлять преимущества и недостатки отдельных технических решений и вычислительных систем в целом.

При решении задачи повышения производительности вычислительных систем возникают два вопроса: несоответствие быстродействия центральной части ЭВМ быстродействию внешних запоминающих устройств и устройств ввода-вывода, а также несоответствие производительности мощных вычислительных систем «локальной» форме их эксплуатации.

Технические задачи повышения производительности. Процесс решения задачи на ЭВМ можно разделить на две логически обособленные части: ввод-вывод информации и непосредственные вычисления по заданным алгоритмам. Логика реализации процесса обработки данных требует, чтобы время решения задачи, приходящееся на ввод и вывод информации, было существенно меньше или не влияло на время счета, поскольку время ввода-вывода с точки зрения использования ЭВМ может рассматриваться в некотором смысле как непроизводительное. Обычно для ЭВМ второго поколения при решении задач время, затрачиваемое на операции ввода-вывода, составляло до 60—80% общего времени решения. При этом вычислительная система практически полностью «монополизировалась» вводом-выводом и задачи пользователей в течение этого времени не выполнялись. Такая ситуация в процессе решения задач была обусловлена характеристиками работы периферийных устройств ЭВМ и принципами реализации операций ввода-вывода для этих машин. В процессе взаимодействия периферийных устройств с центральной частью ЭВМ скорость выполнения операций ввода-вывода определялась менее быстродействующими устройствами — периферийными. Это существенно снижало производительность вычислительных машин.

Для ЭВМ третьего поколения указанное несоответствие компенсируется путем совмещения во времени выполнения операций ввода-вывода для многих периферийных устройств с работой центрального процессора. Кроме того, важную роль здесь играет режим мультипрограммирования, когда в оперативной памяти ЭВМ одновременно находятся несколько программ и процессор «переключается» в ходе работы с выполнения одной программы на выполнение другой [4], [17], [19]. При таком режиме работы ЭВМ в процессе решения задач достигается определенная независимость ее центральной части от периферийных устройств. Тем самым обеспечивается повышение производительности вычислительной системы в целом. Отметим, что реализация принципов совмещения операций и мультипрограммирования потребовала введения в состав вычислительной машины целого ряда дополнительных средств,

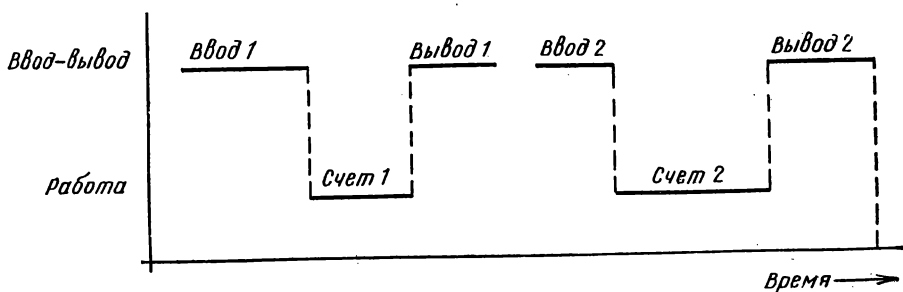


Рис. 40. Временная диаграмма работы ЭВМ в однопрограммном режиме

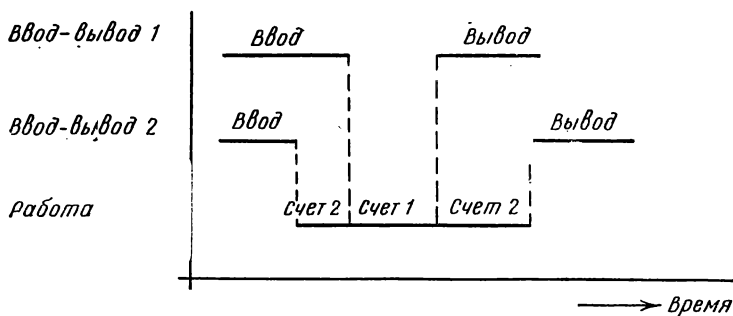


Рис. 41. Временная диаграмма работы ЭВМ в мультипрограммном режиме с двумя задачами (первая имеет более высокий приоритет)

важнейшими из которых являются каналы ввода-вывода [4] и операционные системы [17], [19].

Мультипрограммирование — режим работы вычислительной машины. С позиций эффективного использования ЭВМ и повышения ее производительности принцип мультипрограммирования является одной из существенных особенностей ЕС ЭВМ. Мультипрограммирование, основанное на возможности совмещения операций

ввода-вывода с выполнением программ, реализованной в ЕС ЭВМ, обеспечивает гораздо более высокую загруженность процессора задачами пользователей, чем это наблюдалось ранее. На рис. 40 и 41 показаны временные диаграммы работы ЭВМ в однопрограммном режиме и режиме мультипрограммирования с двумя одновременно находящимися в памяти и выполняемыми программами. В первом случае вычислительная машина может начать работу по второй программе только после полного завершения первой и ввода необходимых данных, включая текст второй программы. Общее время «непроизводительной» работы процессора (T_H) составит в этом случае:

$$T_H = T_{\text{ввод1}} + T_{\text{вывод1}} + T_{\text{ввод2}} + T_{\text{вывод2}}. \quad (4)$$

Для ЭВМ, работающей в режиме мультипрограммирования, ввод первой и второй программы может быть совмещен и, кроме того, возможно совмещение работы одной программы с выводом результатов работы другой. Общее время «непроизводительной» работы процессора здесь составит *:

$$T_H = T_{\text{ввод1}} + T_{\text{вывод2}}. \quad (5)$$

Реализация в ЕС ЭВМ принципа мультипрограммирования потребовала не только разработки новых аппаратных и программных средств, но и существенных изменений в принципах работы традиционных устройств. Практически большая часть особенностей ЕС ЭВМ прямо или косвенно связана с возможностью работы этих машин в мультипрограммном режиме. Пользователям, которые хотят понять и оценить то новое, что дают ЭВМ третьего поколения, необходимо обратить особое внимание на принцип мультипрограммирования и его влияние. Это обеспечит базу для понимания цели и значения новых средств и особенностей, которые имеются в ЕС ЭВМ.

Реализованный в вычислительной машине принцип мультипрограммирования обеспечивает высокий уровень загруженности основных устройств ЭВМ. Практически при одинаковом быстродействии процессоров вычислительная машина, работающая в режиме мультипрограммирования, будет иметь существенно большую производительность, чем ЭВМ, работающая в однопрограммном режиме. В этой связи можно говорить о режиме мультипрограммирования как режиме работы вычислительной машины. Отметим, что режим мультипрограммирования является основным режимом работы для всех моделей ЕС ЭВМ [4].

* В формулах не учтено время подготовки машины к выполнению тех или иных операций. В первом случае — это время установки носителей информации, необходимых второй программе. Во втором случае — время работы специальных средств ЭВМ, обеспечивающих мультипрограммный режим. Влияние этих операций на общее время работы здесь не рассматривается, поскольку целью данного изложения является иллюстрация особенностей мультипрограммного режима работы, а не детальное исследование процессов функционирования и эксплуатации ЭВМ.

Организационно-экономические задачи повышения производительности ЭВМ. Экономические исследования развития вычислительной техники показали высокую эффективность разработки мощных производительных вычислительных систем по сравнению с системами средней и малой производительности. Этот факт количественно характеризуется «законом Гроша»*, утверждающим, что быстродействие вычислительных машин растет пропорционально квадрату их стоимости. В условиях мультипрограммирования, когда быстродействие процессора может быть использовано с высокой эффективностью, ЭВМ с более высоким быстродействием будет соответственно и более производительной. Этот закон, естественно, не является строгим правилом, верным для сравнения любых ЭВМ, принадлежащих различным классам и поколениям, однако он проявляется как тенденция в развитии вычислительной техники и может быть с успехом применен к анализу машин одного типа.

Таким образом, с позиций разработки и производства более мощные ЭВМ в ряде случаев представляются более экономичными и перспективными. Однако важно иметь в виду, что эксплуатация таких машин одним пользователем зачастую становится уже неэффективной, так как средние объемы и сложность задач, решаемых в СОД отдельных пользователей, обычно не требуют применения высокопроизводительных вычислительных систем и, следовательно, часть их вычислительной мощности просто не будет использована. Задача решается путем совместной эксплуатации подобных вычислительных систем группами пользователей, т. е. созданием систем коллективного пользования. При этом возникает ряд технических вопросов, связанных с методами разделения вычислительных ресурсов ЭВМ между отдельными пользователями. Их решение требует дальнейшего (после мультипрограммирования) развития режимов работы ЭВМ, которое обеспечило бы возможность создания систем коллективного пользования.

5. 1. 2. Организационно-экономические вопросы повышения производительности труда программиста

С появлением языков программирования и трансляторов для них, а также ряда отладочных и сервисных программ резко возросла эффективность работы программистов. Однако в процессе отладки программ, который занимает значительную часть времени от их разработки, программист утратил непосредственный доступ к вычислительной машине. Ранее (на ЭВМ предшествующих поколений) при отладке программы программист работал непосредственно за пультом управления вычислительной машины, прослеживая ход решения задачи. Это позволяло оперативно локализовать и устранить большое число ошибок и ускоряло процесс отладки новых программ. Однако подобный способ эксплуатации ЭВМ являлся неэффективным с позиций использования вычислительных средств, поскольку, вмешиваясь в ход выполнения программы, про-

* См., например, Communications of the ACM, 1966, vol. 9, № 6, p. 435—440.

граммист непроизводительно тратил важнейший ресурс вычислительной системы — время работы процессора. Это вызвано тем, что время реакции программиста в ходе отладки программы в десятки, а то и в сотни тысяч раз больше, чем время реакции ЭВМ. Таким образом, можно сказать, что в процессе отладки вычислительная машина работала непроизводительно. Поэтому режим эксплуатации ЭВМ, при котором программист имел непосредственный доступ к вычислительным средствам (режим непосредственного доступа), постепенно заменялся режимом пакетной обработки *. Термин «пакетная обработка» определяет режим эксплуатации ЭВМ, при котором несколько прикладных программ объединяются в группу, называемую пакетом, последовательно вводятся в ЭВМ и выполняются. Пакеты регулярно подготавливаются обслуживающим персоналом системы обработки данных обычно таким образом, чтобы среднее время обработки одного пакета составляло 6—8 ч. Результаты выполнения прикладных программ пакета пользователи получают лишь по окончании обработки всего пакета.

При этом режиме эксплуатации эффективность использования вычислительных средств гораздо выше, чем при непосредственном доступе, поскольку ЭВМ в пакетном режиме работают автономно с минимальным необходимым вмешательством оператора. Однако время отладки программ в пакетном режиме существенно увеличивается, что в свою очередь снижает производительность труда программиста. Даже выявление простых ошибок, возникающих при подготовке программы и исходных данных к ней на носителях (перфокарты, перфоленты), может потребовать нескольких дней при пакетном режиме эксплуатации ЭВМ.

Решение этой задачи может быть достигнуто путем разделения вычислительных ресурсов ЭВМ между несколькими программистами, когда каждый из них имеет непосредственный доступ к машине. В этом случае удается обеспечить быстрый ответ всем программистам. С другой стороны, подобный режим эксплуатации ЭВМ позволяет более эффективно использовать вычислительные средства, чем в случае непосредственного доступа одного программиста.

5. 1. 3. Значение режимов работы и эксплуатации ЭВМ

Таким образом, попытка решения задач, описанных выше, приводит к необходимости анализа человеко-машинной системы как целого и поиска путей повышения ее производительности. При этом отметим, что те или иные решения, призванные повысить эффективность человеко-машинной системы, могут не обеспечивать оптимальные значения производительности ЭВМ и программистов. Как видно из изложенного, требования повышения производительности человеко-машинной системы могут быть удовлетворены двумя способами.

* Отметим, что термины «пакетная обработка» и «пакет прикладных программ» характеризуют различные понятия и никак не связаны между собой.

Во-первых, это организация определенных режимов эксплуатации ЭВМ, не связанных с какими-либо изменениями в технических средствах и программном обеспечении. Фактически некоторые режимы эксплуатации определяют характер взаимодействия пользователя с вычислительной машиной вне зависимости от программно-аппаратной реализации самой ЭВМ. Примером такого режима является пакетная обработка, рассмотренная выше. Этот режим может применяться в процессе эксплуатации вычислительных машин, имеющих как однопрограммный, так и мультипрограммный режим работы.

Во-вторых, определенные требования пользователей могут быть реализованы только путем организации режимов эксплуатации, которые базируются на применении специальных программных и аппаратных средств ЭВМ. В этом случае говорят о режиме работы ЭВМ, обеспечивающем данные режимы эксплуатации. Примером может служить режим параллельной пакетной обработки программ. Здесь так же, как и при обычном пакетном режиме, в работу вводится пакет (группа) программ, но уже с обязательным условием мультипрограммной работы ЭВМ. При параллельной обработке каждой прикладной программе из группы выделяется определенный квант времени для ее выполнения, по прошествии которого управление передается следующей программе. Таким образом, все программы пакета выполняются не последовательно, а параллельно. Преимуществом такого режима эксплуатации является возможность получения результатов по коротким программам до окончания обработки всего пакета. Режим работы ЭВМ с мультипрограммированием и средствами квантования времени * [17] называется режимом с квантованием времени. Отметим, что режим с квантованием времени, эффективный с позиций человеко-машинной системы, не является оптимальным с позиций производительности вычислительной машины.

5. 2. РЕЖИМ ТЕЛЕОБРАБОТКИ

5. 2. 1. Вычислительные системы и режим телеобработки

Выше было определено, что решение задач загрузки высокопроизводительных ЭВМ и увеличение производительности труда программиста приводят к необходимости создания вычислительных систем, в которых существует возможность одновременного использования ресурсов ЭВМ различными пользователями. Такие режимы эксплуатации ЭВМ носят название «режимы разделения времени».

В большинстве случаев при использовании ЭВМ в режимах разделения времени желательно, чтобы различные пользователи

* Средства квантования времени — средства ЭВМ, обеспечивающие прерывание работы процессора через заданные промежутки времени, равные фиксированному значению кванта времени.

вычислительной системы имели доступ к ней со своих рабочих мест, т. е. чтобы их присутствие на вычислительном центре не являлось необходимостью. При организации подобных режимов эксплуатации предъявляются определенные требования к принципам построения и функционирования вычислительных систем. Наиболее существенные из них определяются необходимостью передачи данных на расстояние. В процессе передачи данных обычно используются телефонные и телеграфные линии связи.

Режим работы, при котором удаленные пользователи связываются с ЭВМ, используя телефонные, телеграфные или специальные каналы связи, называется режимом телеобработки. Данный режим в большинстве случаев служит базой для реализации режимов разделения времени. В этом смысле режим телеобработки представляет режим работы ЭВМ, обеспечивающий эффективность человеко-машинной системы. Рассмотрим принципы работы ЭВМ Единой системы в этом режиме подробнее.

Систему телеобработки можно представить как вычислительную систему, реализованную аппаратными и программными средствами и расширенную средствами связи и передачи данных. Важными компонентами вычислительной системы в этом случае являются удаленные абонентские пункты (АП), соединенные с ЭВМ линиями связи. Абонентские пункты в режиме телеобработки представляют средства общения пользователей с вычислительной системой [4]. Основной единицей передачи данных от АП к ЭВМ и обратно является сообщение. В зависимости от характера решаемой задачи сообщение может представлять запрос на обработку данных определенной программой, находящейся в составе библиотек программ вычислительной системы, и данные, передаваемые на ЭВМ или другой АП, и т. п.

Некоторые вопросы организации режима телеобработки. Создание системы телеобработки связано с необходимостью решения ряда задач, которые для традиционных вычислительных систем не возникают.

Одна из таких задач определяется тем, что длины сообщений, передаваемых в системе телеобработки по линиям связи, могут существенно различаться. Короткие сообщения, содержащие всего несколько байт информации, должны обрабатываться одновременно с длинными сообщениями, имеющими тысячи байт. При этом средства телеобработки и программно-аппаратные средства ЭВМ должны быть достаточно гибкими для эффективного приспособления к подобным изменениям в процессе работы.

Другая задача обусловлена тем, что поступление сообщений в систему от удаленных абонентских пунктов в большинстве применений является случайным и непрогнозируемым. Аналогично требования ко времени обработки, размеру оперативной и внешней памяти и занятость линий связи определяются работой удаленных пользователей и не могут быть спланированы заранее вычислительной системой. Чтобы компенсировать трудности, обусловленные этими особенностями, средства телеобработки и программно-

аппаратные средства ЭВМ также должны иметь значительную гибкость, обеспечивая выделение необходимых ресурсов каждой выполняемой программе без полной их монополизации той программой, которая выполняется в данный момент. В результате этой особенности систем телеобработки возникает задача устранения пиковых нагрузок. В некоторые моменты времени число запросов, получаемых системой с удаленных АП, может превышать возможности ЭВМ. Для обеспечения нормальной работы системы необходимо в процессе ее проектирования моделировать условия будущей работы с тем, чтобы привести в соответствие мощность центральной ЭВМ с числом АП, подключаемых к ней, и другими характеристиками системы [50].

Кроме того, целый ряд специфических задач организации систем телеобработки обусловлен особенностями передачи данных по линиям связи. Надежность передачи сообщений по линиям связи существенно зависит от различных условий окружающей среды: близко расположенных высоковольтных линий, электрических импульсов, вызванных молнией, проводимых невдалеке электросварочных работ и др. Из-за этих явлений возникают ошибки в передачах. Обнаружение и коррекция этих ошибок особенно важны в системах телеобработки и требуют специального внимания при создании таких систем.

5. 2. 2. Принципы построения системы телеобработки

Структура и основные компоненты системы телеобработки. Вычислительная система, работающая в режиме телеобработки, может быть разделена на отдельные подсистемы (рис. 42) [4].

Подсистема обработки данных, включающая центральный процессор и оперативную память, предназначена для обработки сообщений и выполнения программ управления системой телеобработки.

Подсистема управления и накопления данных служит для хранения библиотек программ и массивов данных вычислительной системы, а также промежуточного хранения очередей сообщений, предназначенных для обработки или передачи удаленным АП.

Подсистема управления вводом-выводом данных обеспечивает совмещение операций ввода-вывода от удаленных АП и местных устройств ввода-вывода с работой процессора. Заметим, что АП могут располагаться в непосредственной близости от ЭВМ и не подключаться к линиям связи, в этом случае они называются локальными АП в отличие от удаленных АП, подключаемых к линиям связи. Локальные АП могут быть удалены от процессора на расстояние 400—600 м и соединены с каналом ЭВМ обычным кабелем.

Эти три подсистемы в системе телеобработки могут эксплуатироваться в пакетном режиме пользователями, находящимися непосредственно на вычислительном центре и использующими обычные устройства ввода вывода.

Подсистема удаленного доступа системы телеобработки содержит средства передачи данных по каналам связи, средства общения пользователя с ЭВМ, а также специальные средства управления каналами связи. Рассмотрим более подробно основные элементы подсистемы удаленного доступа.

Абонентский пункт состоит из устройства управления и одного или нескольких устройств ввода-вывода. Каждое из этих средств называется компонентом данного АП. Пользователь вводит данные

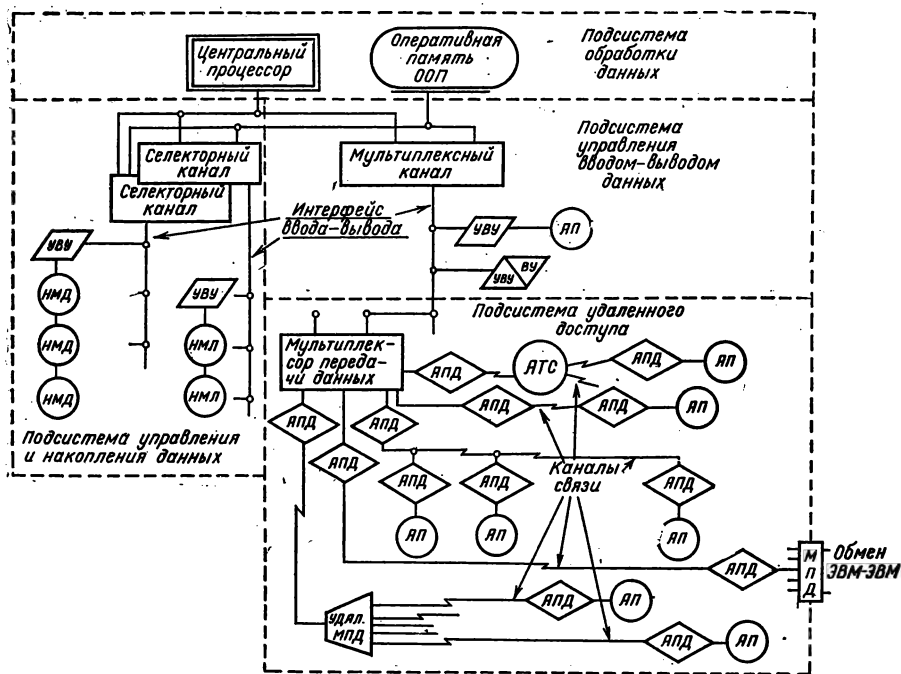


Рис. 42. Комплекс технических средств системы телеобработки

с устройства ввода, которыми оснащен АП (с клавиатуры, перфокарт, перфолент, магнитной ленты). Результаты, поступающие от ЭВМ, могут быть выведены на пишущую машинку, АЦПУ, перфоленту, магнитную ленту или электронно-лучевую трубку (дисплей) [4], [50].

Все сообщения, поступающие от АП и обратно, должны пройти через ЭВМ. Кроме того, вычислительная система может сама генерировать и передавать сообщения для АП.

Абонентский пункт связан с ЭВМ через мультиплексор передачи данных, аппаратуру передачи данных и каналы или линии связи.

По принципу организации связи между устройствами в системе линии связи делятся на некоммутируемые и коммутируемые. Некоммутируемая линия связи соединяет устройства постоянно без переключения. Такая линия называется однопунктовой, если она

соединяет единственный АП с ЭВМ, и многопунктовой, если к ней подключено более одного АП. Коммутируемая линия связи устанавливает связь между ЭВМ и удаленной установкой путем набора номера. Подобно тому как это происходит в телефонной сети, действительный путь для передачи автоматически выбирается из числа возможных путей средствами коммутации. Коммутируемые линии всегда однопунктовые, так как в процессе любого вызова ЭВМ связана только с одной установкой.

Сеть телеобработки может быть коммутируемой, некоммутируемой или комбинированной. Вызов может осуществляться либо от центральной ЭВМ, либо от удаленного АП, а вызванное устройство может ответить либо автоматически, либо с помощью оператора.

Не все возможности систем телеобработки доступны для любых конфигураций линий связи и удаленных АП. В пределах ограничений, устанавливаемых программно-аппаратными средствами телеобработки, пользователь может выбрать возможности, наилучшим образом удовлетворяющие его задачам.

Мультиплексор передачи данных (МПД) представляет собой устройство сопряжения ЭВМ с каналами связи [4], [50]. Логически МПД обеспечивает управление удаленной периферией системы телеобработки. МПД, реализованные в настоящее время в ЕС ЭВМ, выполняют следующие функции: поочередный опрос подключенных каналов связи; последовательно-параллельное формирование и расформирование принимаемых и передаваемых данных; защиту от ошибок при передаче данных, согласование сигналов между ЭВМ и аппаратурой передачи данных; реализацию процедур управления по командам от ЭВМ, включая обмен информацией с ЭВМ. Остальные функции по управлению передачей данных возложены на каналы ввода-вывода (мультиплексный канал) и центральный процессор и обеспечиваются с помощью операционных систем и методов доступа, реализуемых в операционных системах.

Основным элементом аппаратуры передачи данных (АПД), связывающей МПД и удаленные АП с линиями связей, является модем [4], [50]. Модем обеспечивает связь между установкой и линией, преобразуя двоичные сигналы оконечных устройств обработки данных в частоты передачи оборудования связи и обратно. Обычно при использовании развитого программного обеспечения телеобработки программист непосредственно не связан с модемами.

Принцип функционирования системы телеобработки. Для описания принципа функционирования системы телеобработки и определения взаимодействия указанных подсистем рассмотрим последовательность процедур управления по обработке одного сообщения с последующей передачей ответного сообщения удаленному АП. Предполагается, что входное сообщение разбито на сегменты и каждый сегмент отсылается в область оперативной памяти, имеваемую буфером.

В подсистеме удаленного доступа с помощью МПД осуществляется опрос подсоединенных через каналы связи АП. Если появился запрос на передачу сообщения, он через подсистему управления

вводом-выводом поступает в подсистему обработки данных с обращением к управляющей программе. Далее следует указание подсистемам удаленного доступа и ввода-вывода начать передачу. Одновременно в подсистеме обработки данных в оперативной памяти осуществляется выделение буфера. Указанный буфер заполняется сегментом сообщения, и по заполнении сегмент пересылается во внешнюю память подсистемы управления и накопления данных. Далее таким образом пересылаются другие сегменты до тех пор, пока не будет передано все сообщение. После этого осуществляется выборка и пересылка из внешней памяти в основную программы, необходимой для обработки сообщения. Сообщение обрабатывается с возможными обращениями во внешнюю память. Результат обработки также помещается во внешнюю память, и производится обращение к подсистемам удаленного доступа и управления вводом-выводом с указанием начать вывод. Одновременно, как и при вводе, выделяется буфер в оперативной памяти. Далее последовательно осуществляется пересылка сегмента ответного сообщения в буфер, передача его удаленному АП и так до тех пор, пока все сообщение не будет передано по адресу абонентского пункта.

Рассмотренная последовательность процедур управления, в которой участвуют все подсистемы телеобработки, несколько идеализирована в том смысле, что каналы связи, АП, каналы ввода-вывода, центральный процессор, основная и внешняя память считаются доступными при необходимости в любой момент. На самом деле подобный режим полной доступности в любое время крайне неэкономичен. За тот же самый промежуток времени через систему телеобработки проходят другие сообщения. Различные сообщения будут совместно использовать общий фонд ресурсов. При этом необходимо организовать очереди запросов на совместное распределение и использование ресурсов подсистем телеобработки. В связи с этим в системе появятся новые функции планирования и контроля. Эти функции особенно важны для телеобработки, где в отличие от обычных систем обработки данных усложняющими факторами являются те особенности, о которых говорилось выше.

5. 2. 3. Программные средства телеобработки

В реализации функций управления важную роль играет программное обеспечение системы телеобработки. При этом специальные средства программного обеспечения для управления телеобработкой не только позволяют многим пользователям совместно эксплуатировать технические ресурсы вычислительной системы, но и использовать общий набор прикладных программ. Это особенно удобно в тех случаях, когда система ориентируется на определенную сферу применения и многие пользователи с ее помощью решают однотипные задачи. Примером подобной системы может служить вычислительный центр, обслуживающий ряд АСУ производственных предприятий одной отрасли или обеспечивающий сбор основных отчетных материалов для предприятий различных отраслей по единым формам документов.

Такое применение вычислительных систем является эффективным, однако при этом к системам обработки данных предъявляются определенные дополнительные требования. В частности, программы, эксплуатируемые несколькими пользователями, должны учитывать специфику оборудования, которым располагает каждый из них, поскольку эта специфика (различные виды АП и АПД) отражается на характере общения удаленного АП с ЭВМ и, следовательно, на самих программах. В условиях, когда число коллективно используемых прикладных программ может быть достаточно велико, а число пользователей и, следовательно, различных видов оборудования может постоянно возрастать, организация программ с учетом специфики каждого пользователя вряд ли оправдана. Гораздо более эффективный путь решения проблемы состоит в создании унифицированных программных средств, обеспечивающих связь прикладных программ с любыми средствами пользователей.

Одна из наиболее совершенных программных реализаций, обеспечивающих функционирование систем телеобработки, представляет собой комплекс программ, объединенных под названием «общий телекоммуникационный метод доступа». Эти программы входят в состав операционной системы ОС ЕС и обеспечивают функционирование системы телеобработки при работе ЭВМ в режиме мультипрограммирования с фиксированным или переменным числом задач [17].

Основная функция программного обеспечения телеобработки состоит в управлении передачей сообщений между абонентскими пунктами или между абонентскими пунктами и прикладными программами пользователей (рис. 43). При этом следует различать задачи управления сетью телеобработки, управления очередями сообщений и редактирования сообщений*.

При управлении сетью программы общего телекоммуникационного метода доступа устанавливают связь и управляют движением сообщений между центральной ЭВМ и АП, выделяют, используют и освобождают буферы, требуемые для хранения сообщений в процессе обработки. Кроме того, в процессе управления сетью необходимо обнаруживать ошибки, исходящие от АП или линий связи, и по возможности их устранять или извещать о них пользователей. Обычно при обнаружении ошибок рассматриваемые программы телеобработки автоматически производят повторную передачу сообщений. В случае идентификации неустранимой ошибки сообщение о ней посылается оператору системы.

Управление очередями сообщений представляет чрезвычайно важную задачу при организации систем коллективного пользования, требующих эффективного разделения ресурсов. Различные алгоритмы организации и обслуживания очередей определяют и различия во времени обработки тех или иных сообщений, а также различия в использовании других ресурсов системы. Например,

* Миллз Д. Л. Программное обеспечение систем передачи данных. — В сб.: Системы передачи данных и сети ЭВМ. Под ред. П. Грина и Р. Лаки. (Пер. с англ.). М., «Мир», 1974.

в тех случаях, когда время является наиболее важным ресурсом для некоторых пользователей системы, очереди сообщений, поступающих от них (высокоприоритетных сообщений), и ответных сообщений могут храниться не во внешней, а в основной памяти. Благодаря этому существенно сокращается время реакции системы на такие запросы.

В соответствии с требованиями программы общего телекоммуникационного метода доступа помещают сообщения в очереди к указанным прикладным программам и абонентским пунктам на устройство прямого доступа или в основную память. При этом могут учитываться приоритеты сообщений.

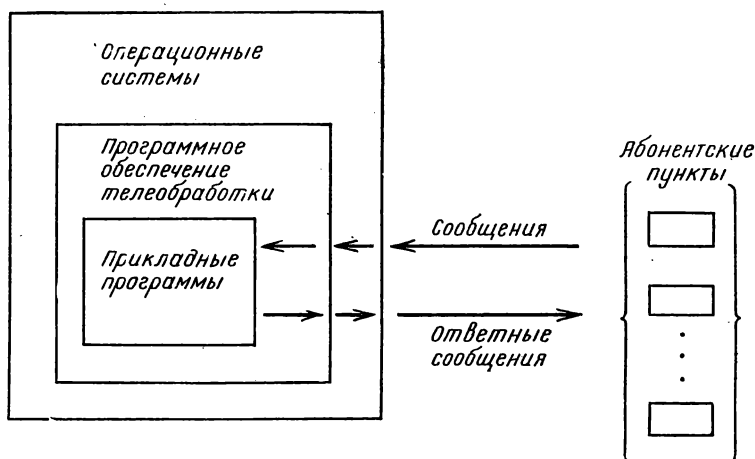


Рис. 43 Роль программного обеспечения телеобработки

Всякое сообщение в системе телеобработки состоит из заголовка и основного текста [50]. Заголовок представляет собой управляющую часть сообщения, в которой содержатся сведения о коде сообщения, его приоритете, типе и т. п. Текст сообщения определяет содержание той информации, которая передается либо для обработки на ЭВМ, либо как результат обработки на АП.

При получении сообщений программы телеобработки должны расшифровать его заголовок и обработать текст сообщения в соответствии с полученной информацией. Аналогично по окончании обработки и перед передачей ответного сообщения его текст должен быть снабжен новым заголовком. Решение подобных задач относится к области редактирования сообщений. Кроме того, в процессе редактирования сообщений осуществляются трансляции кодов сообщений (если рабочие коды АП и ЭВМ не совпадают), удаление символов, «зачеркнутых» пользователем в процессе подготовки сообщений при работе с клавиатурой и т. п.

Общий телекоммуникационный метод доступа представляет набор макрокоманд для управления сообщениями. Практически эти средства образуют гибкий специализированный язык высокого

уровня. Макрокоманды программ телеобработки могут быть использованы для построения ориентированной на конкретную вычислительную систему программы управления сообщениями, которая управляет потоком сообщений между АП или между АП и прикладными программами.

Программа управления сообщениями разрабатывается на языке макрокоманд общего телекоммуникационного метода доступа программистами, обслуживающими данную сеть телеобработки. При этом учитывается специфика каналов связи, аппаратуры передачи данных, абонентских пунктов и прочих устройств, применяемых в системе. Функционально макрокоманды языка управления сообщениями разделяются в соответствии с задачами, рассмотренными выше, на макрокоманды управления сетью телеобработки, управления очередями сообщений и редактирования сообщений. Выбирая подходящие макрокоманды и их операнды, программист определяет логические модули программ телеобработки, которые войдут в данную программу управления сообщениями, а также действия, которые эти модули должны выполнять. При этом он может построить систему в полном соответствии с требованиями предстоящих применений.

Общий телекоммуникационный метод доступа функционирует под управлением операционной системы ОС ЕС. Программа управления сообщениями и прикладные программы телеобработки могут выполняться как отдельные задания. Программа управления сообщениями может быть инициализирована и выполнена в любом разделе оперативной памяти. Однако она должна выполняться как задание с самым высоким приоритетом и поэтому может эффективно работать только в разделе (для режима мультипрограммирования с фиксированным числом задач) или зоне (для режима мультипрограммирования с переменным числом задач) с наивысшим приоритетом. Прикладные программы работают параллельно с управляющей программой и могут быть выполнены в любом из оставшихся разделов или любой зоне. Супервизор операционной системы управляет асинхронной работой различных программных компонентов системы.

Обработка сообщений может вестись с помощью только одной прикладной программы или в ситуации, когда различные сообщения требуют различных программ. В первом случае программа обработки может быть добавлена к программе управления сообщениями. Она обеспечит обработку всех типов сообщений и при необходимости прекратит работу управляющей программы. В случае применения одной программы для обработки любых видов сообщений существует возможность повышения эффективности системы телеобработки путем постоянного хранения данной программы в оперативной памяти (рис. 44а). Здесь при получении системой нового сообщения не требуется дополнительного времени на ввод программы с целью его обработки. При этом сама программа обработки может быть построена таким образом, что она никак не изменится в процессе обработки каждого сообщения и поэтому одна

ее копия может одновременно использоваться для обработки нескольких сообщений. Это свойство программ называется реентерабельностью [17].

Изложенный метод построения и использования прикладных программ в системе телеобработки особенно эффективен при жест-

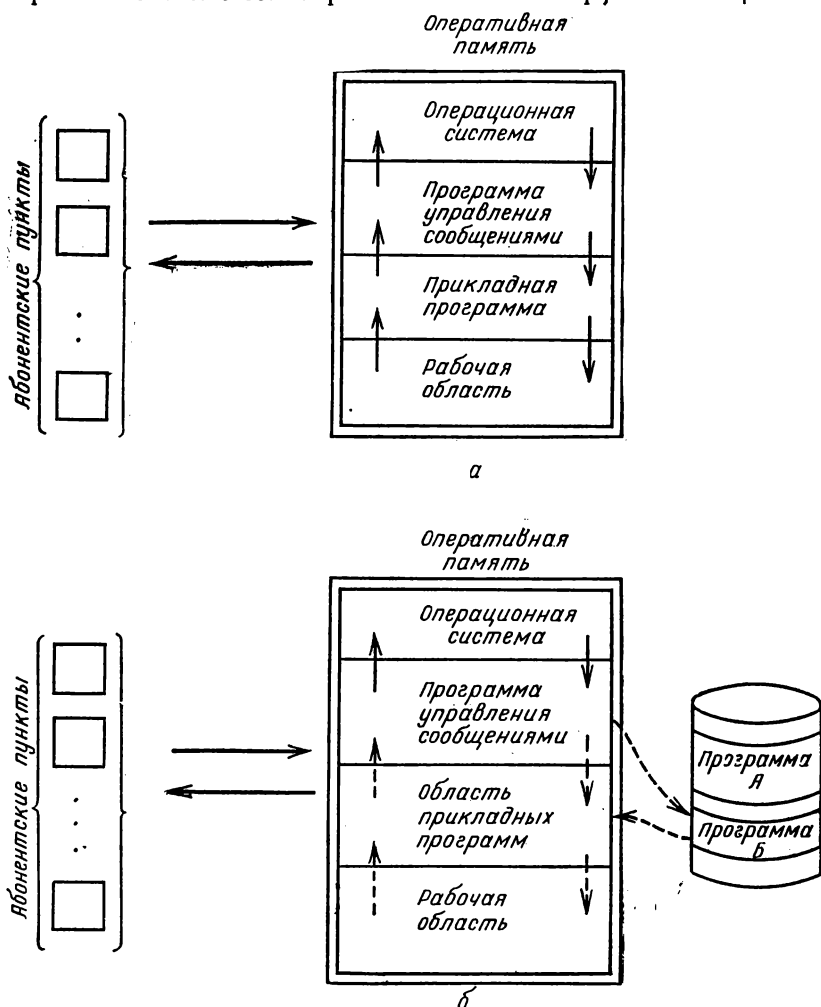


Рис. 44. Варианты работы системы телеобработки:
а — в системе с одной программой обработки всех видов сообщений;
б — в системе с различными программами обработки различных видов сообщений

ких ограничениях на время ответа. В частности, он применяется в системах резервирования билетов на авиалиниях.

Другой возможный вариант состоит в обработке сообщений различных типов с помощью различных прикладных программ. В этом

случае программы обработки сообщений могут располагаться на внешних носителях информации и вводиться в оперативную память по мере необходимости (рис. 446). Однако не исключается и совместное использование этого метода с изложенным выше. Например, программы обработки высокоприоритетных сообщений могут быть сделаны реентерабельными и постоянно находиться в оперативной памяти. Эффективность того или иного варианта построения системы должна специально оцениваться в каждом конкретном случае, исходя из специальных требований сферы применения и конкретных пользователей.

Функции программного обеспечения телеобработки при работе с прикладными программами. Прикладные программы, используемые в системе телеобработки, применяют макрокоманды общего телекоммуникационного метода доступа для получения данных и передачи результатов абонентским пунктам. Программы телеобработки обеспечивают такое сопряжение с прикладными программами, которое позволяет программисту обрабатывать сообщения из сети телеобработки такими же простыми макрокомандами, как и при обработке последовательных данных для местных устройств ввода-вывода. Поскольку для выполнения операций передачи сообщений используется программа управления сообщениями, прикладная программа может быть независимой от устройств передачи. Однако для правильного построения той части программы, которая управляет передачей сообщений, программист должен иметь общее представление о концепциях управления каналом связи.

Таким образом, функции общего телекоммуникационного метода доступа для системы телеобработки можно разделить на две категории: управление сообщениями с помощью программы управления сообщениями и обработка сообщений прикладными программами. В первом случае программа управления сообщениями выступает по отношению к вычислительной системе как высокоприоритетная программа, работающая под управлением операционной системы. Во втором — программа управления сообщениями представляет промежуточное звено между удаленными АП и прикладными программами обработки сообщений.

Программы телеобработки, реализующие описанные функции, могут быть использованы самостоятельно, либо служить базой для реализации более совершенных режимов эксплуатации ЭВМ. Основные формы применения общего телекоммуникационного метода доступа как самостоятельного средства следующие: сбор данных; коммутация сообщений; обработка справочных запросов.

Сбор данных. Сбор данных может быть полностью возложен на программу управления сообщениями и не требует применения специальных прикладных программ. Удаленные АП посылают данные на ЭВМ в форме сообщений, которые накапливаются и запоминаются вычислительной машиной с целью последующей обработки. Программа управления сообщениями может собирать данные двумя способами: запоминанием данных в очередях для прикладных программ и запоминанием данных на любом внешнем носителе.

ле информации. При использовании первого способа сообщения могут быть получены прикладной программой в любое время, для этого необходимо обратиться к данной программе за обработкой. Подобная обработка может, например, осуществляться периодически в течение дня. При втором способе сообщения последовательно записываются на выбранное устройство внешней памяти. В дальнейшем эти данные должны обрабатываться без применения программных средств телеобработки.

Коммутация сообщений. Коммутация сообщений также полностью осуществляется программой управления сообщениями и не требует специальных прикладных программ. При коммутации сообщений удаленные АП передают сообщение вычислительной машине, которая, в свою очередь, посылает эти сообщения одному или более АП. При этом допускается обработка данных сообщений с помощью прикладных программ одновременно с выполнением коммутации или позже.

Обработка справочных запросов. Справочные запросы обрабатываются с помощью программы управления сообщениями и специально разработанных программ. Программа управления сообщениями принимает запросные сообщения и запоминает их в буферах, подготавливая к обработке соответствующей программой. Эта программа обрабатывает информацию, заключенную в сообщениях, и передает ответ программе управления сообщениями. Последняя, в свою очередь, направляет это сообщение соответствующему АП. Программа, вызываемая для обработки сообщения, не обязательно должна находиться в основной памяти. Например, в системах, где обрабатывается много запросов различных типов, неэкономично иметь все требуемые программы обработки в основной памяти. В этом случае может быть разработана специальная программа анализа, которая определит тип сообщения и вызовет соответствующую программу обработки. Подобная программа анализа может быть объединена с программой управления сообщениями.

Значение программных средств телеобработки. Рассмотренные программные средства телеобработки и функции, выполняемые ими в системе, в той или иной форме присутствуют в любой реализации распределения ресурсов вычислительной системы между большим числом удаленных пользователей*. В этом смысле данные программные средства вместе с комплексом специализированного оборудования представляют базу для функционирования вычислительных систем в режиме телеобработки. Таким образом, несмотря на то что средства телеобработки находят самостоятельные применения, часть из которых описана выше, можно сказать, что данный режим представляет собой режим работы вычислительной системы.

* Мы не говорим здесь о таких средствах, как базисный телекоммуникационный метод доступа, который в силу ограниченности выполняемых им функций может быть использован либо в небольших по масштабам применениях, либо как база для создания систем рассмотренного выше типа,

5. 3. РЕЖИМЫ РАЗДЕЛЕНИЯ ВРЕМЕНИ

Основная задача, определяющая необходимость поиска новых режимов эксплуатации вычислительных систем, состоит в том, чтобы, с одной стороны, обеспечить высокую степень загруженности вычислительной системы и всех ее технических и программных компонентов, а с другой — оперативность получения результатов обработки данных пользователями. Анализ существовавших до настоящего времени режимов эксплуатации показывает, что для удовлетворения обоим критериям никакие организационные методы эксплуатации уже не являются достаточными как в случае непосредственного доступа, так и режима пакетной обработки данных. Необходимый результат может быть достигнут лишь путем создания вычислительной системы, в которой несколько пользователей могли бы одновременно разделять имеющиеся ресурсы. Для реализации данной задачи нужны дополнительные технические и программные средства, образующие совместно с традиционными средствами обработки данных системы телеобработки. На базе этих систем удастся реализовать новые режимы эксплуатации ЭВМ — режимы разделения времени. Рассмотрим подробнее эти режимы и их основные особенности.

Режимы разделения времени. Под режимами разделения времени мы подразумеваем различные формы эксплуатации систем телеобработки, содержащих несколько АП, которые могут работать одновременно, или формы эксплуатации вычислительных систем, имеющих локальные абонентские пункты *.

Число различных режимов разделения времени достаточно велико, и они определяют некоторые особенности работы системы. Эти различия обусловлены несколькими факторами, важнейшими из которых являются требования к оперативности получения результатов обработки и сложность решаемых задач. Хотя на практике в «чистом виде» эти режимы редко встречаются, рассмотрим их отдельно с целью определения особенностей каждого.

Удаленная пакетная обработка. Наиболее простым режимом эксплуатации системы телеобработки является режим, при котором данные, поступающие с удаленных АП, устанавливаются в очередь и обрабатываются с помощью программ, находящихся в библиотеках, расположенных на внешних носителях информации в вычислительном центре. Результаты обработки посылаются адресату на требуемый абонентский пункт. При этом после отсылки данных пользователь практически не участвует в процессе решения задачи. В этом случае время решения не ограничивается короткими интервалами, однако не ограничивается и сложность решаемой задачи. Фактически при таком режиме эксплуатации ЭВМ функции системы телеобработки состоят лишь в приеме и передаче данных без каких-либо специальных дополнительных требований ко времени

* Первый случай представляется более общим, и поэтому в дальнейшем рассматриваются в основном системы телеобработки.

решения задачи. Обычно программы, предназначенные для работы в режиме пакетной обработки, после небольших модификаций могут быть использованы и для удаленной пакетной обработки. При этом модификация должна обеспечить связь данной программы с программой управления сообщениями для передачи результатов обработки требуемому абонентскому пункту.

Запросно-ответный режим. Эксплуатация системы телеобработки в этом режиме была описана выше при рассмотрении общего телекоммуникационного метода доступа и наиболее простых применений программы управления сообщениями. При данном режиме эксплуатации вычислительной машины достигается оперативность получения пользователем результатов обработки данных, особенно в тех случаях, когда для удовлетворения запроса не нужны крупные и сложные программы обработки. Поэтому запросно-ответный режим применяется в первую очередь при решении достаточно простых задач, если требуется оперативное получение результатов обработки. Примером таких систем могут служить система резервирования билетов на авиалиниях, некоторые информационные системы в АСУ. В частности, таким образом могут быть реализованы задачи кадрового учета, когда любая требуемая информация оформляется в виде запроса, реализация которого осуществляется программой обработки данного запроса и результат обработки предоставляется пользователю в виде выходных форм или отчетов.

Диалоговый режим. Данный режим эксплуатации систем телеобработки является наиболее сложным с позиций реализации, однако он может обеспечить решение сложных задач в условиях ограниченного времени. Внешне процесс решения задачи в диалоговом режиме протекает таким образом, что пользователь и вычислительная система регулярно обмениваются информацией, уточняющей и направляющей ход решения. Диалоговый режим широко применяется в различных сферах, и его значение при переходе от одной сферы применения к другой может существенно изменяться. Например, диалоговый режим может использоваться программистом для составления и отладки программ. В этом случае функцией диалога является контроль правильности составления программы непосредственно в ходе ее разработки.

Другой вариант применения диалоговых систем может быть обусловлен необходимостью доступа к системе широкого пользователя, не только не являющегося специалистом в области программирования, но и не знакомого с той программой, которую ему предстоит использовать. В этом случае диалог ведет сама программа, определяя каждый раз не только свой ответ на очередной вопрос пользователя, но и предлагая ему возможные варианты дальнейших вопросов. Примерами таких систем могут служить информационно-поисковые системы для ведения каталогов, библиографий в крупных библиотеках и др. [47].

Третий возможный вариант применения диалоговых систем предполагает полуавтоматизированное решение сложных задач в ситуации, когда в определенных узловых пунктах решения выбор

дальнейшего направления может быть предоставлен пользователю. Примером могут служить системы в АСУ. В частности, при решении задач оперативного управления с использованием методов СПУ пользователь может применить диалоговую систему, позволяющую оперативно обрабатывать сетевые графики небольшого объема, получать необходимую информацию по результатам обработки, оперативно производить коррекции сетевых графиков и т. д.

Приведенные примеры не исчерпывают всех возможностей применений диалоговых систем, однако дают представление о широких перспективах таких систем и сложности их разработки.

С точки зрения оперативности решения задачи эти системы обычно обеспечивают реализацию ответов на отдельные запросы пользователей в требуемые сроки. Однако следует учесть, что в силу сложности решаемых задач, которая здесь выше, чем в системах типа «запрос-ответ», время реакции системы в данном случае может также быть несколько выше при одном и том же числе пользователей системы. Обычно такие допущения возможны в тех случаях, когда оперативность обработки данных не является важным требованием системы. С другой стороны, сложность решаемых задач может быть значительно выше, чем в системе типа «запрос-ответ».

Работа в реальном масштабе времени. До сих пор при рассмотрении режимов эксплуатации систем телеобработки анализировался характер взаимодействия пользователя с ЭВМ. Однако в некоторых случаях применения таких систем важнейшими критериями являются: время реакции системы и число пользователей, эксплуатирующих ее, для которых время реакции является решающим фактором. Системы, в которых данные критерии являются основными и где решение в заданное время, обусловленное содержанием самой задачи, является важнейшим требованием, называются системами, работающими в реальном масштабе времени.

Характерными примерами систем такого типа могут служить автоматизированные системы управления технологическими процессами (АСУТП), где вычислительные машины используются для регулирования работы станков, конвейеров и т. п.*. Здесь сложность используемых программ относительно невысока, однако требования к времени реакции системы являются предельно жесткими, а число управляемых объектов может измеряться сотнями. Другим примером из области взаимодействия пользователей с вычислительной машиной в реальном масштабе времени является система резервирования и распределения билетов на авиалиниях в случае, когда число пользователей имеет тот же порядок и при этом абонентские пункты могут быть расположены в различных городах страны. При этом требования к времени реакции системы не являются столь жесткими, как в предыдущем случае.

* Эти задачи могут и не требовать средств телеобработки в тех случаях, когда удаление управляемых объектов находится в заданных пределах. Однако важнейшие проблемы управления сообщениями, которые рассматривались выше, решаются и здесь.

В системах, работающих в реальном масштабе времени и взаимодействующих с пользователями, оперативность обработки данных и возможность обслуживания большого числа абонентов достигаются за счет относительной простоты и малого числа эксплуатируемых программ. Таким образом, рассматривая систему, работающую в реальном масштабе времени, с позиций ее взаимодействия с пользователем, можно констатировать ее близость к системам типа «запрос-ответ», хотя вряд ли такие системы должны всегда совпадать.

Смешанные режимы эксплуатации. Реальные системы телеобработки обычно используются для обеспечения различным пользователям возможности работы в различных режимах эксплуатации. При этом все перечисленные выше режимы могут «смешиваться» различными способами в произвольных пропорциях. Иногда для реализации смешанных режимов эксплуатации необходимы дополнительные технические, а чаще программные средства, управляющие работой пользователей в различных режимах. Например, возможно совмещение режима работы в реальном масштабе времени и удаленной пакетной обработки на одной и той же вычислительной системе. Такое совмещение режимов удобно в том смысле, что позволяет загрузить вычислительную систему задачами даже при отсутствии в определенные периоды времени запросов с жесткими ограничениями по времени. Однако при реализации подобного совмещения возникают дополнительные задачи планирования процесса вычислений, автоматического перемещения программ и данных между основной и внешней памятью и др. Поскольку не все функции такого рода реализуются существующими телекоммуникационными методами доступа, появляется необходимость создания дополнительных программных средств для обеспечения таких режимов. Подобные программные средства объединяют в пакет программ, называемый монитором реального времени. Для различных вычислительных систем могут быть использованы различные мониторы, отличающиеся требованиями, предъявляемыми к вычислительной системе, а также объемом и характером выполняемых функций. По структуре монитор реального времени представляет собой программную систему, работающую как высокоприоритетная программа под управлением основной операционной системы. Отметим, что монитор реального времени, как и большинство пакетов прикладных программ, расширяющих возможности основных операционных систем, не требует каких-либо изменений в операционных системах, кроме, может быть, специальной генерации.

Смешанные режимы используются в большинстве случаев в системах коллективного пользования, где отдельные абоненты могут иметь или не иметь общие задачи и соответственно общее программное обеспечение. В отличие от подобных систем можно рассматривать специализированные вычислительные системы, предназначенные для решения четко определенных задач, общих для всех эксплуатирующих ее пользователей. К таким системам можно отнести, например, уже рассмотренную ранее информационно-поисковую систему для библиотечного хозяйства.

С позиций разработки смешанные системы обычно предъявляют более жесткие требования экономического характера, чем специализированные системы. Это обусловлено тем, что в системах коллективного пользования в условиях решения различного рода не связанных между собой задач эффективность систем должна определяться вне зависимости от той или иной сферы применения, а в первую очередь как функция загрузки вычислительной системы при условии абсолютного удовлетворения требований отдельных пользователей. Напротив, в специализированных системах требование оптимальной загрузки вычислительной системы хотя и является важным, однако может отойти на второй план, поскольку эффективность системы определяется в первую очередь самой сутью решаемых задач (например, системы резервирования мест на авиалиниях).

Одной из наиболее важных задач в области проектирования вычислительных систем коллективного пользования, связанных с необходимостью повышения их эффективности, является задача выбора конфигурации комплекса технических средств и операционной системы, оптимальных для обслуживания определенной группы пользователей с определенным набором задач [50].

5. 4. АРХИТЕКТУРА ПРИКЛАДНЫХ ПРОГРАММ И РЕЖИМЫ РАЗДЕЛЕНИЯ ВРЕМЕНИ

Анализируя прикладные программы, используемые в системе телеобработки, важно отметить, что в отличие от программ, работающих в пакетном режиме, здесь структура программ обуславливается в основном не сферой применения, а тем или иным режимом эксплуатации системы телеобработки. В зависимости от ряда факторов в одних и тех же сферах применения могут использоваться различные режимы разделения времени.

Особый интерес представляет тот факт, что архитектуры программ для режимов разделения времени аналогичны архитектурам программ, используемых в пакетном режиме, несмотря на очевидные особенности первых, обусловленные спецификой работы системы телеобработки. Рассмотрим различные архитектуры программ и режимы, в которых они применяются.

Библиотеки программ. Данная структура программ характерна для эксплуатации систем телеобработки в режиме «запрос-ответ». Здесь каждая программа библиотеки предназначена для обработки данных по определенному запросу. При этом вызов программы может осуществляться таким образом, что пользователь, посылая запрос, может и не знать, какая именно программа будет вызвана для обработки, и прочие детали процесса обработки. В этом смысле библиотеки программ для систем типа «запрос-ответ» являются более сложными, чем библиотеки для пакетной обработки. Это особенно важно при использовании данного режима эксплуатации в АСУ, где от пользователя не требуется специальных знаний при общении с вычислительной системой.

К недостаткам систем подобного рода можно отнести относительную простоту решаемых задач. Эта простота обусловлена требованием оперативности обработки данных, которая не всегда может быть обеспечена для сложных задач, реализованных в виде отдельных программ библиотеки. Кроме того, в автоматизированных системах управления это привело бы к возникновению крупных «монолитных» программ, трудно поддающихся модификациям. Таким образом, не обеспечивая решения всех возможных задач в системах телеобработки, библиотеки программ эффективно применимы для реализации оперативной обработки данных в достаточно простых ситуациях. Если учесть, что в процессе работы программы библиотеки могут обращаться к банкам данных универсального типа или специально ориентированным на работу в режиме «запрос-ответ», становится очевидным, что область применения систем такого рода может быть достаточно широкой и обусловлена именно необходимостью оперативного получения информации в режиме разделения времени. В противном случае, например, для задач АСУ более эффективно использование программных систем, где в рамках одного пакета прикладных программ в результате глубокой модульности системы можно сосредоточить решение большого числа задач с меньшим дублированием алгоритмов и программ. В случае реализации решения сложных задач библиотекой программ в АСУ такое дублирование почти неизбежно.

Специализированные (диалоговые) системы программирования с использованием языков. Цель подобной формы программирования — возможность разработки программ и их отладки в процессе непрерывного взаимодействия с вычислительной системой. В этом смысле программы данной архитектуры позволяют обеспечить одновременно нескольким программистам режим «непосредственного» доступа к ЭВМ без ущерба для ее загруженности, т. е. преодолеть одну из наиболее существенных проблем. Таким образом, функции специализированных языков программирования в системах телеобработки существенно отличаются от функций обычных специализированных языков. Подобная форма программирования широко распространена в системах телеобработки и внешне по принципам разработки программ мало отличается от работы с обычными языками высокого уровня. В обоих случаях язык программирования предназначен для описания алгоритма решения задачи, обеспечивающего после отладки и трансляции решение задачи на данной вычислительной машине.

В процессе работы с диалоговым языком программирования программист осуществляет пооператорный ввод программы. Система производит синтаксический анализ каждого оператора и в случае появления ошибок извещает о них программиста. После внесения исправлений в описание данного оператора и повторного контроля его правильности процесс может быть продолжен. После полного введения программы производится попытка ее выполнения. При этом могут быть обнаружены семантические ошибки,

сообщения о некоторых из них также выдаются программисту. После внесения соответствующих исправлений и синтаксического контроля программа вновь выполняется [52]*. Таким образом, система в данном случае в полной мере совмещает функции обычного транслятора для языка программирования высокого уровня и эффективного отладочного средства.

В настоящее время известно большое число языков программирования с трансляторами для работы в диалоговом режиме. При этом разработчики программного обеспечения создавали, с одной стороны, соответствующие трансляторы для известных ранее широко распространенных языков, с другой — реализовывали новые языки, специально предназначенные для работы в системах телеобработки. В качестве примера первого варианта можно привести разработанный фирмой IBM диалоговый транслятор с языка Фортран (в настоящее время подобные трансляторы с языка Фортран поставляются не только этой фирмой). Примерами специально разработанных языков могут служить БЕЙСИК и АПЛ.

Диалоговые языки программирования представляют собой в первую очередь средства, предназначенные для программиста. Однако такой язык, как БЕЙСИК [52], настолько прост в изучении, что может использоваться для проведения несложных, но трудоемких вычислений широким пользователем, например, в рамках АСУ предприятия. Основы подобного языка можно изучить в короткий срок, и при этом не требуется серьезного знания принципов работы вычислительных систем и построения программ. Таким образом, здесь диалоговый язык программирования может рассматриваться как вспомогательное средство для проведения трудоемких вычислений с использованием ЭВМ в условиях отсутствия специальных программ для решения данных задач.

Программные системы. Помимо использования режима диалога для разработки программ существуют и некоторые другие важные сферы его применения. Поскольку диалог при этом не связан с разработкой программ, пользователь в этих случаях обменивается с вычислительной системой данными, подлежащими обработке или определяющими необходимый вид обработки. В соответствии с полученными данными программа, реализующая диалог, автоматически генерирует процесс обработки. Подобный принцип работы характерен для программ, имеющих структуру программных систем. Поэтому практически в сфере телеобработки данных использование диалоговых режимов работы, не связанных с разработкой новых программ, реализуется обычно с помощью программных систем. Рассмотрим некоторые типичные задачи, при решении которых эффективно применение диалоговых систем.

* Такого рода системы относятся к системам компилирующего типа. Наряду с ними существуют интерпретирующие системы, которые обеспечивают не только пооператорный контроль, но и пооператорное выполнение программы. Однако такие системы применяются обычно лишь для решения достаточно простых задач, так как предъявляют слишком жесткие требования к построению программы.

Одной из важнейших сфер применения диалоговых систем является класс задач, подразумевающих обеспечение связи с вычислительной системой широкого пользователя. В этом случае диалоговая система должна не только вести обработку необходимых данных, но и на каждом этапе диалога предлагать пользователю возможные пути его дальнейшего развития. Практически логика работы подобных систем может быть представлена древовидной структурой, где вершина представляет собой начало диалога, а каждое разветвление — возможные варианты его дальнейшего развития. Очевидно, в таком представлении сложность программной системы зависит от объема и сложности самого дерева диалога, а также от сложности обработки данных в промежутках между вершинами.

Поскольку режим диалога представляет естественные требования к оперативности обработки данных, части программ, реализующие обработку данных между двумя соседними вершинами, не должны быть слишком объемными. Оптимальное построение дерева диалога должно удовлетворять определенным требованиям по времени проведения всего диалога в целом, а также по времени реализации отдельных этапов. Подобные системы широко применяются при разработке информационных систем, в библиотечном деле при диагностике неисправностей сложных технических систем, а также для программированного обучения пользователей [50]. В качестве примера последнего применения можно привести программу Learn ATS/360, специально разработанную фирмой IBM для обучения пользователей работе с программой ATS/360 (Administrative Terminal System).

Другой важной сферой применения диалоговых систем является класс задач, в которых диалог инициируется пользователем, а не программой. В этом случае по окончании каждого этапа диалога пользователь самостоятельно указывает, какой этап должен быть выбран следующим. В данном случае схема диалога также может быть представлена графически. Однако обычно при этом она бывает несколько сложнее уже потому, что редко имеет четко выраженную древовидную структуру. Здесь, в частности, в ходе диалога могут зачастую встречаться ситуации, когда пользователю необходимо возвратиться к более ранним этапам диалога с тем, чтобы затем продолжить обработку с новыми данными на базе анализа предшествующих. Подобные диалоговые системы могут получить широкое распространение в АСУ, поскольку обеспечивают пользователю возможности моделирования различных ситуаций, встречающихся в управлении, и принятия решения на основе полученных результатов. При этом подобные диалоговые системы обеспечивают возможность анализа большого числа моделей с использованием различных наборов исходных данных в короткий срок. Примером такой системы может служить пакет прикладных программ MINIPERT, разработанный фирмой IBM для расчета временных характеристик сетевых графиков. Эта программа используется для небольших по объему сетей (до 200 работ в одной подсети при

объеме рабочей области памяти 32К). При этом пользователь на языке программной системы может затребовать или осуществить ввод данных, описание данных и их модификацию, а также различные виды обработки с получением различных форм выходных отчетов. Общее число операторов языка системы, используемых для обращения к ней, около 20. Программа MINIPERT позволяет не только выполнять необходимые вычисления по сетевым графикам, но и обеспечивает возможность моделирования альтернативных вариантов плана реализации проекта наглядно и в короткий срок.

Благодаря указанным преимуществам диалоговые системы второго типа эффективно используются в АСУ наряду с системами типа «запрос-ответ» и режимом удаленной пакетной обработки.

ПРИЛОЖЕНИЕ 1

НЕКОТОРЫЕ РЕЗУЛЬТАТЫ АНАЛИЗА ХАРАКТЕРИСТИК ПАНЕТОВ ПРИКЛАДНЫХ ПРОГРАММ И СФЕР ИХ ПРИМЕНЕНИЯ

В приложении приведены статистические данные по результатам анализа отечественных и зарубежных прикладных программ. Описания этих программ сосредоточены в ряде каталогов, предназначенных для пользователей. Общее число рассмотренных программ около 200.

Анализ прикладных программ помог выявить некоторые тенденции разработки и эксплуатации программного обеспечения, которые сформулированы в книге. В связи с этим полученные статистические данные, на наш взгляд, имеют определенное самостоятельное значение и служат для иллюстрации изложенного материала.

Прошло еще не так много времени с тех пор, как модели ЕС ЭВМ поступили к пользователю и приобрели значительный удельный вес в общем парке вычислительных машин, поэтому еще не накоплено достаточно опыта широкой эксплуатации ЭВМ этого семейства. Однако необходимо проводить систематическую работу в направлении анализа их эксплуатации. Это обеспечит эффективность применения современных вычислительных средств. Такая работа должна проводиться как отдельными пользователями в рамках конкретных систем обработки данных, так и централизованно. В этой связи систематизацию некоторых часто применяемых прикладных программ можно рассматривать как одно из направлений анализа эксплуатации вычислительных машин.

Все рассмотренные программы в соответствии с их распространенностью у пользователей и уровнем программной реализации разделены на два типа. Это — прикладные программы и обеспечивающие программы.

Виды классификации программ, встречающиеся в таблицах (по сферам применения, ориентации и т. п.), подробно описаны в книге и не требуют специальных пояснений. Каждая таблица сопровождается комментариями, в которых кратко формулируются результаты анализа.

Методы вычислений.

В некоторых случаях возникают затруднения при попытке однозначно идентифицировать программы в соответствии с принятой в таблицах классификацией. Это имеет место при классификации программ по сферам применения, языкам программирования и архитектуре.

Классификация по сферам применения. Некоторые программы при классификации могут быть отнесены к различным сферам применения. Так, например, встречаются программы, в одинаковой мере относящиеся и к программам, расширяющим возможности операционных систем, и к программам общего назначения.

Классификация по языкам программирования. Одна и та же программа часто разрабатывается с применением нескольких языков программирования. При этом различные программные модули разрабатываются на разных языках с учетом специальных требований к их выполнению.

Классификация программ по архитектуре. В основном рассмотренные программы удается однозначно классифицировать по архитектуре.

Однако в некоторых случаях одна и та же программа может иметь черты программной системы и системы программирования с использованием специализированного языка. В этом случае последняя форма программирования используется как вспомогательное средство для настройки системы на требования пользователя.

В тех случаях, когда по определенной классификации некоторые программы не идентифицировались однозначно, они учитывались отдельно для каждого класса, к которому их можно было отнести. Таким образом, одна программа может быть учтена в таблице дважды (реже — трижды). Это несколько затрудняет расчет, однако обеспечивает возможность более точно отразить тенденции в создании прикладных программ.

Для большей наглядности статистических данных и для обеспечения возможности сравнения характеристик различных классов были применены методы нормирования полученных результатов вычислений, с тем чтобы сумма процентных отношений для программ в рамках той или иной классификации равнялась 100%.

Пояснения к табл. 1.

Задачи общего назначения. Данные в таблице показывают, как с повышением уровня программной реализации уменьшается число программ общего назначения. Это обусловлено тем, что прикладные программы общего назначения создавались на основе обеспечивающих программ, реализованных различными пользователями. Так, в частности, количество обеспечивающих программ, предназначенных для реализации методов линейного и целочисленного программирования, — 10, в то время как в составе рассмотренных прикладных программ 5 программ по своим возможностям практически перекрывают все эти обеспечивающие программы.

Задачи по расширению возможностей операционных систем. Реализация программ, расширяющих возможности операционных систем, требует от программистов высокой квалификации и хорошего знания этих систем. Практическая работа по созданию таких программ может эффективно проводиться лишь разработчиками операционных систем. В связи с этим программы, расширяющие возможности операционных систем, сосредоточены в основном в составе прикладных программ. Доработки и модификации этих программ с целью обеспечения их гибкости и универсальности, а также с целью устранения ошибок проводятся, как и для операционных систем путем регулярного выпуска новых, более совершенных версий.

Задачи экономики и управления. Анализ показывает, что в этой области состав задач, решаемых прикладными и обеспечивающими программами, почти одинаков. Этим, в частности, объясняется относительно равномерное распределение программ в данной сфере применения по типам.

Совершенствование программ этой сферы применения проводилось в основном в направлении обеспечения большей гибкости программных реализаций, а также в направлении создания комплексов взаимосвязанных пакетов прикладных программ, реализующих решение задач управления в рамках интегрированных систем обработки данных. Удельный вес взаимосвязанных пакетов прикладных программ в данной сфере применения — 48%, а для обеспечивающих программ — 13%.

Научные и инженерные задачи. Большинство обеспечивающих программ в этой сфере применения не имеет аналогов в составе прикладных программ. В этом смысле обеспечивающие программы служат не только базой для программных реализаций более высокого уровня, но и существенно расширяют состав задач, решаемых прикладными программами. Это обусловлено непрерывным возникновением в данной сфере применения новых прикладных задач.

Пояснения к табл. 2.

Данные в таблице иллюстрируют широкое распространение машинно-ориентированного языка Ассемблера при реализации прикладных программ и важную роль языков высокого уровня для обеспечивающих программ. Практически во всех сферах применения (кроме научных и инженерных) использование языков высокого уровня при разработке прикладных программ носит вспомогательный характер. Основная часть программ разработана на Ассемблере.

В научных и инженерных применениях ЭВМ, где преобладают программы

однократного использования и высокая эффективность программной реализации не является важнейшим требованием, широкое распространение получает Фортран. Преимуществами языка Фортран являются простота в использовании и ориентация на применение в научных и инженерных областях. Эти преимущества определяют массовое применение Фортрана при создании обеспечивающих программ.

Однако удельный вес использования языка Ассемблера и для обеспечивающих программ остается достаточно высоким в тех случаях, когда при создании программ требовалась эффективность функционирования вычислительной системы (для программ, расширяющих возможности операционных систем), или в связи с преобладанием программ регулярного использования (для программ, реализующих задачи экономики и управления).

В последнее время проявляется тенденция к широкому распространению языка ПЛ/1, обладающего большими возможностями. Это, однако, не нашло отражения в таблице, поскольку данные в ней фиксируют состояние с разработкой и эксплуатацией прикладных программ на определенный момент времени. Небольшой удельный вес применения языка ПЛ/1 объясняется, с одной стороны, тем, что этот язык появился сравнительно недавно и ещё недостаточно освоен пользователями. С другой стороны, применение этого языка несколько сложнее, чем применение языков Фортран, Кобол и др., поскольку ПЛ/1 не только универсален с позиций использования, но и отражает важнейшие особенности ЭВМ третьего поколения и их операционных систем.

Анализ пакетов прикладных программ. Анализ рассмотренных пакетов прикладных программ выявил следующее их распределение по сферам применения, ориентации, архитектуре и средствам разработки, %:

Сфера применения:		Архитектура:	
задачи общего назначения	21	программная система	74
задачи по расширению возможностей операционных систем	25	система программирования*	23
задачи экономики и управления	33	библиотека программ	3
научные и инженерные задачи	12	Языки программирования:	
прочие задачи	9	Ассемблер	70
Ориентация:		Фортран	16
		ПЛ/1	8
		Кобол	5
		прочие	1
методо-ориентированные	16		
проблемно-ориентированные	54		
неклассифицированные	30		

* Здесь и далее в таблицах вместо термина «система программирования с использованием специализированного языка» используется сокращение — «система программирования».

Сфера применения. Нетрудно заметить, что основную массу пакетов прикладных программ составляют программы для решения задач экономики и управления, расширения возможностей операционных систем и задачи общего назначения. Именно здесь существует возможность создания широкого набора регулярно используемых прикладных программ. Во многих сферах применения ЭВМ постоянно требуется создание программ однократного использования. Базой для создания таких программ служат прикладные программы общего назначения. Этим обусловлено регулярное использование прикладных программ данного класса.

Ориентация. Некоторые пакеты прикладных программ не удается классифицировать по ориентации, поэтому выделяется группа неклассифицированных программ. Основную часть этой группы составляют программы, расширяющие возможности операционных систем.

Приведенные данные показывают, что большая часть пакетов прикладных программ имеет ориентацию на решение определенной проблемы. В этом проявляется тенденция к выделению наиболее важных прикладных задач и их программной реализации в обособленных прикладных программах. Иначе говоря, общее число распространенных прикладных задач в процессе использования ЭВМ растет быстрее, чем количество методов, применяемых для их решения.

Архитектура. Преобладание программной системы, как формы программирования для пакетов прикладных программ обусловлено наличием большого числа проблемно-ориентированных программ, которые в основном разрабатываются именно в этой форме.

Пояснения к табл. 3.

Задачи общего назначения. Поскольку каждая задача общего назначения в любой конкретной сфере применения интерпретируется как содержательная задача пользователя, ее программная реализация с ориентацией на метод более целесообразна. Этим обусловлено преобладание в данной сфере применения методо-ориентированных программ. Наличие большого числа неклассифицированных программ в этой сфере применения связано с тем, что многие программы общего назначения зачастую реализуют задачи по расширению возможностей операционных систем, которые мы не относили ни к методо-, ни к проблемно-ориентированным.

Ориентация программ данной сферы применения на метод обуславливает относительно широкое применение в этой сфере таких форм программирования, как библиотеки программ и системы программирования с использованием специализированных языков. Однако даже в этой сфере программные системы применяются достаточно широко.

Задачи по расширению возможностей операционных систем. Программы в этой сфере применения вообще не классифицировались по ориентации.

Поскольку программная реализация задач этого класса требует высокой степени автоматизации вычислительного процесса, здесь преобладающей формой программирования является программная система. Однако зачастую программы, расширяющие возможности операционных систем, снабжаются специализированным языком программирования, предназначенным для упрощения программной реализации задач определенного класса. Отметим, что в большинстве случаев система программирования с использованием специализированного языка не является здесь самостоятельной формой программирования, а применяется совместно с программной системой. Удельный вес программ, реализованных в форме системы программирования, не объединенной с программной системой, составляет 18% общего числа пакетов, при создании которых использовалась архитектура системы программирования.

Задачи экономики и управления. В связи с тем, что в данной сфере применения основную группу пользователей составляет широкий пользователь, преобладающее значение здесь имеют проблемно-ориентированные программы. Кроме того, отметим, что проблемная ориентация обусловлена здесь и широким распространением программ регулярного применения. Проблемная ориентация и требование высокой степени автоматизации вычислительного процесса обуславливают создание прикладных программ в этой сфере в основном в форме программных систем.

Научные и инженерные задачи. Поскольку научные и инженерные задачи обычно конкретизируются как задачи более узкой сферы применения, например задачи электротехники, механики и др., здесь преобладает проблемная ориентация, хотя процентные соотношения далеко не так показательны, как для задач экономики и управления. Преобладание в данной сфере проблемно-ориентированных программ обуславливает и применение в качестве основной формы программирования программных систем. Отметим, однако, что специализированные языки программирования здесь также находят широкое применение. Так, удельный вес программ, для которых система программирования с использованием специализированного языка является самостоятельной формой программирования, составляет 67% общего числа программ, в которых эта форма использовалась.

Пояснения к табл. 4.

Нетрудно видеть, что хотя прикладные программы, имеющие структуру программной системы, преобладают как для методо-, так и для проблемно-ориентированных программ, тем не менее соотношения числа программных систем и систем программирования с использованием специализированных языков для проблемно-ориентированных программ гораздо выше, чем для программ, ориентированных на метод. Заметим также, что для разработки программ, ориентированных на метод, более широко применяется язык программирования Фортран.

Таблица 1

**РАСПРЕДЕЛЕНИЕ ПРОГРАММ ПО СФЕРАМ ПРИМЕНЕНИЯ
В РАМКАХ ТИПА ПРОГРАММ, %**

Типы программ	Сфера применения				
	задачи общего назначе- ния	задачи по расширению возможностей операционных систем	задачи экономики и управле- ния	научные и инженер- ные задачи	прочие задачи
Прикладные	21	25	33	12	9
Обеспечивающие	33	2	31	21	13

Таблица 2

**РАСПРЕДЕЛЕНИЕ ПРОГРАММ ПО ИСПОЛЬЗОВАННЫМ ЯЗЫКАМ
ПРОГРАММИРОВАНИЯ В РАМКАХ ТИПА ПРОГРАММ И СФЕРЫ ПРИМЕНЕНИЯ, %**

Сфера применения	Прикладные программы					Обеспечивающие программы				
	Ассемблер	Фортран	ПЛ/1	Кобол	прочие	Ассемблер	Фортран	ПЛ/1	Кобол	прочие
Задачи общего назначения	78	16	3	2	1	11	82	7	—	—
Задачи по расширению возможностей операцион- ных систем	85	9	3	2	1	50	50	—	—	—
Задачи экономики и управления	68	6	16	9	1	39	34	9	16	2
Научные и инженерные задачи	41	53	4	—	2	5	73	11	8	3
Прочие задачи	82	5	—	11	2	57	—	23	12	3

Таблица 3

**РАСПРЕДЕЛЕНИЕ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ ПО ОРИЕНТАЦИИ,
АРХИТЕКТУРЕ И ИСПОЛЬЗОВАННЫМ ЯЗЫКАМ ПРОГРАММИРОВАНИЯ
В РАМКАХ СФЕРЫ ПРИМЕНЕНИЯ, %**

Сфера применения	Ориентация			Архитектура			Язык программирования				
	методо- ориентиро- ванные	проблемно- ориентиро- ванные	некласси- фицирован- ные	программная система	система программи- рования	библиотека программ	Ассемблер	Фортран	ПЛ/1	Кобол	прочие
Задачи общего назначения	47	7	46	43	49	8	78	16	3	2	1
Задачи по расши- рению возможно- стей операционных систем	—	—	100	70	28	2	85	9	3	2	1
Задачи экономики и управления	2	83	15	91	7	2	68	6	16	9	1
Научные и инже- нерные задачи	19	69	12	63	37	—	41	53	4	—	2
Прочие задачи	6	94	—	88	12	—	82	5	—	11	2

Таблица 4

**РАСПРЕДЕЛЕНИЕ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ ПО АРХИТЕКТУРЕ
И ИСПОЛЬЗОВАННЫМ ЯЗЫКАМ ПРОГРАММИРОВАНИЯ
В РАМКАХ ОРИЕНТАЦИИ, %**

Ориентация	Архитектура			Язык программирования				
	програм- мная система	система програм- мирова- ния	библио- тека про- грамм	Ассемблер	Фортран	ПЛ/1	Кобол	прочие
Методо-ориентированные	48	42	10	69	24	6	—	1
Проблемно-ориентирован- ные	86	13	1	67	14	12	5	2

Таблица 5

**РАСПРЕДЕЛЕНИЕ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ ПО ОРИЕНТАЦИИ
И ИСПОЛЬЗОВАННЫМ ЯЗЫКАМ ПРОГРАММИРОВАНИЯ
В РАМКАХ АРХИТЕКТУРЫ, %**

Архитектура	Ориентация			Язык программирования				
	методо-ориентированные	проблемно-ориентированные	неклассифицированные	Ассемблер	Фортран	ПЛ/1	Кобол	прочие
Программная система	9	59	32	69	19	8	4	—
Система программирования	26	28	46	79	16	—	3	2
Библиотека программ	50	25	25	40	20	20	17	3

Пояснения к табл. 5.

Данные в таблице показывают, что применение программной системы как формы программирования в гораздо большей степени распространено для проблемно-ориентированных программ, чем для методо-ориентированных. С другой стороны, такие формы программирования, как библиотека программ и система программирования с использованием специализированного языка, в большей мере применяются при разработке методо-ориентированных программ. Отметим, что высокие показатели по применению специализированных языков для разработки проблемно-ориентированных программ во многом обусловлены их использованием совместно с программными системами. Удельный вес программ, имеющих элементы обеих архитектур, от общего числа программ, имеющих архитектуру системы программирования с использованием специализированного языка, составляет для проблемно-ориентированных программ 60%.

ПРИЛОЖЕНИЕ 2

ФОРМЫ ВЫХОДНЫХ ДОКУМЕНТОВ ППП СПУ/ЕС

Выходные отчеты по временным характеристикам сетей и подсетей создаются программами процессора отчетов в результате обработки промежуточного набора данных, передаваемого от программ процессора сетей. Система обеспечивает возможность получения десяти стандартных отчетов как по всей сети, так и отдельно по любым подсетям, причем в последнем случае подсети могут быть обработаны в зависимости от запроса как связанные или как обособленные.

Создание той или иной выходной формы вызывается путем указания номера этой формы (от 01 до 10) в карте запроса отчетов, которая является основным элементом языка процессора отчетов. Выходные формы имеют одинаковый формат (см. таблицу ниже) и различаются видами сортировки выдаваемых данных. Рассмотрим каждый из выдаваемых отчетов подробнее.

ФОРМАТ ВЫХОДНЫХ ОТЧЕТОВ 01—09 ПО ВРЕМЕННЫМ ХАРАКТЕРИСТИКАМ РАБОТ

Код предшествующего события работы	Код последующего события работы	Код группы работ*	Краткое содержание работы	Оценки времени выполнения работы	Первичный резерв времени выполнения работы	Вторичный резерв времени выполнения работы	Ранняя дата завершения работы	Поздняя дата завершения работы	Плановая дата завершения работы	Код подразделения, выполняющего работу

* Код, которым могут быть помечены любые работы сети. Обычно в сетевом графике выделяются по какому-либо признаку интересующие пользователя группы работ и помечаются одним и тем же кодом. При этом коды работ для разных групп должны быть различными. Код группы работ может представлять собой любой набор двух символов, каждый из которых может быть буквой или цифрой. Этот элемент данных может не выдаваться, так как он необходим только для формы № 10.

Сортировка «предшествующее событие — последующее событие» (форма № 01). Отчет представляет информацию о работах в порядке возрастания буквенно-цифровых кодов их начальных событий. Если одно событие является начальным для нескольких работ, эти работы упорядочиваются в отчете в порядке возрастания кодов их конечных событий.

Эта форма наглядно представляет зависимость работ от их предшествующих событий и совместно с формой № 09 полностью отражает структуру сетевого графика.

Сортировка по коду подразделения и ранней дате окончания работы (форма № 02). Отчет представляет информацию о работах в порядке возрастания буквенно-цифровых кодов выполняющих их подразделений. Если одно подразделение выполняет несколько работ, эти работы упорядочиваются в отчете в порядке увеличения ранней даты их окончания.

Этот отчет компонует данные о работах, выполняемых одним подразделением, и совместно с формой № 03 позволяет сформировать календарные планы выполнения работ для каждого подразделения.

Сортировка по коду подразделения и поздней дате окончания работы (форма № 03). Отчет представляет информацию, аналогичную предыдущей, с той разницей, что работы, выполняемые одним подразделением, упорядочиваются по возрастанию поздней даты их окончания.

Совместно с формой № 02 данный отчет позволяет сформировать календарные планы выполнения работ для каждого подразделения.

Сортировка по величине первичного резерва времени выполнения работы (форма № 04). Отчет представляет информацию о работах в порядке возрастания величины первичного резерва времени их выполнения.

В отчете наглядно отражается группа работ, имеющих минимальный первичный резерв. Отчет полезен как для выявления работ, требующих особого внимания (критических и близких к критическим), так и для выявления работ, имеющих большой резерв времени и обеспечивающих гибкость при формировании календарных планов.

Сортировка по ранней дате окончания работы (форма № 05). Отчет представляет информацию о работах в порядке возрастания ранней даты их окончания. Если несколько работ имеют одну и ту же дату окончания, данные о них упорядочиваются по возрастанию кода предшествующего события.

Совместно с формой № 06 данный отчет может быть полезен при формировании календарного плана выполнения всего проекта в целом.

Сортировка по поздней дате окончания работы (форма № 06). Отчет представляет информацию о работах в порядке возрастания поздней даты их окончания. Совместно с формой № 05 данный отчет может быть полезен при формировании календарного плана выполнения всего проекта в целом.

Сортировка по ранней дате окончания работы и коду подразделения (форма № 07). Отчет представляет информацию о работах аналогично форме № 05, с той разницей, что, если несколько работ имеют одну и ту же раннюю дату окончания, данные о них упорядочиваются по возрастанию кода подразделения, ответственного за выполнение работы.

Совместно с формой № 08 данный отчет может служить для эффективного контроля выполнения проекта в различных подразделениях. В частности, скомпонованная в них информация может служить основой для составления программ совещаний по СПУ на предприятии, выполняющем разработку.

Сортировка по поздней дате окончания и коду подразделения (форма № 08). Отчет аналогичен предыдущему, с той лишь разницей, что вместо упорядочения по ранней дате окончания работы происходит упорядочение по поздней дате окончания.

Совместно с формой № 07 данный отчет может служить для эффективного контроля выполнения проекта в различных подразделениях.

Сортировка «последующее событие — предшествующее событие» (форма № 09). Отчет представляет информацию аналогично форме № 01, с той разницей, что при его формировании приоритетным параметром упорядочения является код конечного события работы.

Совместно с формой № 01 данный отчет полностью отражает структуру сетевого графика.

Линейный график хода разработки (форма № 10). Этот отчет иллюстрирует порядок выполнения любых выделенных групп работ. Он отличается по формату от девяти предыдущих отчетов и представляет собой модифицированный линейный график, в котором по горизонтальной оси откладывается время, а по вертикальной следуют коды групп работ*. Факт выполнения работ из группы, помеченной данным кодом, изображается горизонтальным отрезком соответствующей длины, расположенным в строке, отвечающей этому коду, в соответствующем месте по горизонтали. Масштабный коэффициент для временной оси пользователь задает самостоятельно и может выбрать его в достаточно широких пределах.

В рамках описанных стандартных форм отчетов пользователь имеет возможность средствами языка процессора отчетов обеспечить запрос более сжатой и концентрированной информации о ходе выполнения разработки. Так, в частности, отчеты могут содержать данные не обо всех работах, а например, о работах, завершающихся в данном квартале или в данном году; о работах, выполняемых данным подразделением, и т. п. Практически с помощью языковых средств процессора отчетов пользователь может обеспечить выборку информации о работах для отчета по любым полям данных, включаемых в отчет.

* Традиционный график получится, если в качестве кода группы работ использовать код подразделения, выполняющего работу.

ИСПОЛЬЗОВАННАЯ И РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

I. Принципы построения современных вычислительных машин. Некоторые технические средства ЭВМ

1. Байцер Б. Архитектура вычислительных комплексов. (Пер. с англ.). Т. 1 и 2. М., «Мир», 1974.
2. Введение в запоминающие устройства прямого доступа и методы организации данных. Учебное пособие. (Пер. с англ.). М., «Статистика», 1974.
3. Вычислительная система IBM/360. Принципы работы (Пер. с англ.) М., «Сов. радио», 1968.
4. Единая система ЭВМ. Под общей ред. А. М. Ларионова. М., «Статистика», 1974.
5. Катцан Г. Вычислительные машины системы 370. (Пер. с англ.). М., «Мир», 1974.
6. Королев Л. Н. Структуры ЭВМ и их математическое обеспечение. М., «Наука», 1974.
7. Пржиялковский В. В., Смирнов Г. Д., Пыхтин В. Я. Электронная вычислительная машина «Минск-32». М., «Статистика», 1972.
8. Процессор ЭВМ ЕС-1020. Под общей ред. А. М. Ларионова. М., «Статистика», 1975.
9. Флорес А. Организация вычислительных машин. (Пер. с англ.). М., «Мир», 1972.
10. Штойер К.-Х. Электронная обработка данных. Краткие сведения для руководителей. (Пер. с нем.). М., «Статистика», 1973.

II. Принципы построения системного математического обеспечения

11. Алферова З. В., Лихачева Г. Н., Шураков В. В. Математическое обеспечение ЭВМ. М., «Статистика», 1974.
12. Белокурская И. А., Кушнерев Н. Т., Неменман М. Е. Диспетчер ЭВМ «Минск-32». М., «Статистика», 1973.
13. Донован Дж. Системное программирование. (Пер. с англ.). М., «Мир», 1975.
14. Колин А. Введение в операционные системы. (Пер. с англ.). М., «Мир», 1975.
15. Лебедев В. Н. Введение в системы программирования. М., «Статистика», 1975.
16. Мили Г. [и др.]. Функциональная структура OS/360. (Пер. с англ.). М., «Сов. радио», 1971.
17. Система математического обеспечения ЕС ЭВМ. Под общ. ред. А. М. Ларионова. М., «Статистика», 1974.
18. Супервизоры и операционные системы. Под ред. Дж. Каттла и П. Робинсона. (Пер. с англ.). М., «Мир», 1972.
19. Шура-Бура М. Р., Ковалевич Э. В., Марголин М. С. и др. Операционная система ДОС ЕС. Общие положения. М., «Статистика», 1975.

III. Документирование технических средств и программного обеспечения

20. Система документации Единой системы ЭВМ. Под общ. ред. А. М. Лапинова. М., «Статистика», 1975.

21. Уолш Д. Руководство по созданию документации для математического обеспечения. (Пер. с англ.). М., «Наука», 1975.

IV. Прикладное программное обеспечение современных ЭВМ

22. Алгоритмы и организация решения экономических задач. Сб. статей, вып. 6. М., «Статистика», 1975.

23. Джермейн К. Программирование на IBM/360. (Пер. с англ.). М., «Мир», 1971.

24. Мартин Дж. Программирование для вычислительных систем реального времени. (Пер. с англ.). М., «Наука», 1975.

25. Сборник научных программ на Фортране. Статистика. Вып. 1. Матричная алгебра и линейная алгебра. Вып. 2. (Пер. с англ.). М., «Статистика», 1974.

26. Слейгл Дж. Искусственный интеллект. (Пер. с англ.). М., «Мир», 1973.

27. Стэбли Д. Логическое программирование в системе 360. (Пер. с англ.). М., «Мир», 1974.

V. Языки программирования

28. Брудно А. Л. Алгол. М., «Наука», 1971.

29. Васильев В. А. Язык Алгол-68. Основные понятия. М., «Наука», 1972.

30. Коддингтон Л. Ускоренный курс Кобол. (Пер. с англ.). М., «Мир», 1974.

31. Лавров С. С. Универсальный язык программирования. М., «Наука», 1972.

32. Линдси Ч., Мюйлен С. вандер. Неформальное введение в Алгол-68. (Пер. с англ.). М., «Мир», 1973.

33. Первин Ю. А. Основы Фортрана. М., «Наука», 1972.

34. Сэксон Дж. Кобол. (Пер. с англ.). М., «Статистика», 1970.

35. Универсальный язык программирования PL/I. (Пер. с англ.). М., «Мир», 1968.

36. Хигман Б. Сравнительное изучение языков программирования. (Пер. с англ.). М., «Мир», 1974.

VI. Специализированные языки программирования

37. Дал У.-И., Мюрхауг Б., Нюгрод К. СИМУЛА-67. Универсальный язык программирования. (Пер. с англ.). М., «Мир», 1969.

38. Марковиц Г., Хауснер Б., Карр Г. СИМСКРИПТ. Алгоритмический язык для моделирования. (Пер. с англ.). М., «Сов. радио», 1966.

39. Мартин Д. Моделирование на вычислительных машинах. (Пер. с англ.). М., «Сов. радио», 1972.

40. Нейлор Т. [и др.]. Машинные имитационные эксперименты с моделями экономических систем. (Пер. с англ.). М., «Мир», 1975.

VII. Информационные и информационно-поисковые системы

41. Лавров С. С., Гончарова Л. И. Автоматическая обработка данных. М., «Наука», 1971.

42. Ланкастер Ф. У. Информационно-поисковые системы. (Пер. с англ.). М., «Мир», 1972.

43. Лефковиц Д. Структуры информационных массивов оперативных систем. (Пер. с англ.). М., «Энергия», 1973.

44. Мидоу Ч. Анализ информационно-поисковых систем. Введение для программистов. (Пер. с англ.). М., «Мир», 1970.

45. Сэлтон Г. Автоматическая обработка, хранение и поиск информации. (Пер. с англ.). М., «Сов. радио», 1973.

46. Фостер Дж. Обработка списков. (Пер. с англ.). М., «Мир», 1974.

47. Хенли Дж. Автоматизированная библиотека и информационные системы. (Пер. с англ.). М., «Мир», 1974.

VIII. Системы передачи данных и сети ЭВМ

48. Бертэн Ж., Ружие М., Риту М. Работа ЭВМ с разделением времени. (Пер с франц.). М., «Наука», 1972.
49. Мартин Дж. Сети связи и ЭВМ. (Пер. с англ.). Ч. 1. М., «Связь», 1974.
50. Мартин Дж. Системный анализ передачи данных. (Пер. с англ.). Т. 1 и 2. М., «Мир», 1975.
51. Основы современной системотехники. Под ред. М. Рабина. (Пер. с англ.). М., «Мир», 1975.
52. Система «ВАНГ-2200В». Руководство для пользователей. Рига, «Зинатне», 1974.
53. Системы передачи данных и сети ЭВМ. Сб. статей под ред. П. Грина и Р. Лаки. (Пер. с англ.). М., «Мир», 1974.
54. Уилкс М. Системы с разделением времени. (Пер. с англ.). М., «Мир», 1972.

IX. Автоматизированные системы управления

55. Глушков В. М. Введение в АСУ. Изд. 2-е. Киев, «Техника», 1974.
56. Информационные системы — миф и действительность. Сб. статей. (Пер. с англ.). М., «Знание», 1973.
57. Миллер Г. В. ПЕРТ — система управления. (Пер. с англ.). М., «Экономика», 1965.
58. Организационные вопросы автоматизации управления. (Пер. с англ.). М., «Экономика», 1972.
59. Организация управления в фирме «Контрол дейта». (Пер. с англ.). М., «Прогресс», 1974.
60. Расчет экономической эффективности внедрений ЭВМ. (Пер. с японск.). М., «Прогресс», 1968.

ОГЛАВЛЕНИЕ

Введение	3
Глава 1. Сферы применения и характеристики прикладных программ	6
1.1. Состав и функции системы программного обеспечения ЭВМ	6
1.1.1. Значение программного обеспечения ЭВМ	6
1.1.2. Состав системы программного обеспечения	7
1.1.3. Развитие системы программного обеспечения	10
1.2. Пакеты прикладных программ. Основные понятия и характеристики	12
1.2.1. Характеристики сфер применения ЭВМ	15
1.2.2. Квалификация пользователей ЭВМ	16
1.2.3. Характеристики пакетов прикладных программ	18
1.3. Основные формы программирования задач обработки данных	24
1.3.1. Архитектура прикладных программ	24
1.3.2. Свойства программной системы как формы программирования	28
1.3.3. Соотношение различных форм программирования	31
1.4. Архитектура прикладных программ и сферы их применения	33
1.4.1. Пакеты прикладных программ, расширяющие возможности операционных систем	33
1.4.2. Пакеты прикладных программ общего назначения	34
1.4.3. Пакеты прикладных программ для решения научно-технических задач	35
1.5. Некоторые особенности прикладного программного обеспечения автоматизированных систем управления	36
1.5.1. Типизация прикладных программ АСУ. Программная совместимость	36
1.5.2. Типизация прикладных программ АСУ. Гибкость программных реализаций	42
Глава 2. Прикладные программы в системе обработки данных	47
2.1. Некоторые проблемы создания интегрированных систем обработки данных	47
2.1.1. Ранние концепции создания систем обработки данных в АСУ	48
2.2. Централизация данных в системе обработки данных для АСУ	52
2.2.1. Банк данных — центральное звено интегрированной системы обработки данных	52
2.2.2. База данных	54
2.2.3. Система управления базой данных	55

2.2.4. Языковые средства систем управления базой данных	57
2.2.5. Преимущества применения банков данных в системах обработки данных	61
2.2.6. Некоторые вопросы разработки и эксплуатации баз данных	63
2.3. Системы обработки данных в некоторых других сферах применения	65
2.3.1. Информационно-поисковые системы	65
2.3.2. Универсальные системы управления базой данных и передачей данных	67
2.3.3. Запросно-ответные системы	68
2.4. Требования систем организации данных к вычислительным средствам	69

Глава 3. Некоторые вопросы разработки и внедрения пакетов прикладных программ	73
3.1. Разработка программ. Этап технического задания	74
3.1.1. Выработка альтернативных решений	75
3.2. Разработка программ. Этап технического проекта	77
3.2.1. Разработка блок-схем алгоритмов	77
3.2.2. Организация исходных данных	79
3.2.3. Выбор конфигурации технических средств	81
3.2.4. Определение конфигурации операционной системы	84
3.2.5. Планирование модификаций	87
3.2.6. Разработка методов отладки	83
3.3. Разработка программ. Программирование и документирование	90
3.4. Внедрение и сопровождение программных систем	95

Глава 4. Пакет прикладных программ СПУ/ЕС как пример программной системы	99
4.1. Основные сведения о сетевых методах планирования и управления	100
4.1.1. Управление разработками сложных систем	100
4.1.2. Методы сетевого планирования и управления	101
4.2. Возможности ППП СПУ/ЕС и его требования к вычислительным средствам	110
4.2.1. Характеристики обрабатываемых сетей	110
4.2.2. Возможности обработки сетевых графиков	112
4.2.3. Выходная информация	115
4.2.4. Требуемая конфигурация вычислительных средств	115
4.3. Структура и принципы функционирования ППП СПУ/ЕС	116
4.3.1. Принципы построения ППП СПУ/ЕС	116
4.3.2. Структура процессора сетей	119
4.3.3. Структура процессора отчетов	122
4.3.4. Языковые средства для работы с ППП СПУ/ЕС	123
4.4. Языковые средства ППП СПУ/ЕС как основа его эксплуатации	126
4.4.1. Языковые средства процессора сетей	126
4.4.2. Задачи описания данных	127
4.4.3. Языковые средства процессора отчетов	130
4.4.4. Сравнение языковых средств пакета с алгоритмическими языками программирования	130
4.5. Вопросы установки и применения ППП СПУ/ЕС	132
4.5.1. Привязка программы к конфигурации вычислительных средств	132
4.5.2. Привязка программы к специфике задач пользователя	134
4.5.3. Совершенствование программной системы	139
4.5.4. Документация ППП СПУ/ЕС	140

Глава 5. Прикладные программы и режимы эксплуатации вычислительных машин	143
5.1. Режимы работы и режимы эксплуатации ЭВМ	144
5.1.1. Производительность вычислительной машины	144
5.1.2. Организационно-экономические вопросы повышения производительности труда программиста	148
5.1.3. Значение режимов работы и эксплуатации ЭВМ	149
5.2. Режим телеобработки	150
5.2.1. Вычислительные системы и режим телеобработки	150
5.2.2. Принципы построения системы телеобработки	152
5.2.3. Программные средства телеобработки	155
5.3. Режимы разделения времени	162
5.4. Архитектура прикладных программ и режимы разделения времени	166
Приложение 1	171
Приложение 2	176
Литература	179

А. Е. Фатеев, А. И. Ройтман, Т. П. Фатеева

**Прикладные программы в системе
математического обеспечения ЕС ЭВМ**

Редактор *Т. А. Петрова*

Техн. редактор *Р. Н. Феоктистова* Корректоры *Т. М. Васильева,*
З. С. Кандыба

Худ. редактор *Т. В. Стихно*

Переплет художника *Е. И. Алексеева*

Сдано в набор 28/І 1976 г. Подписано к печати 14/VI 1976 г.

Формат бумаги 60×90¹/₁₆ Бумага № 3 Объем 12,82 печ. л.

Уч.-изд. л. 12,82 Усл. п. л. 11,5 Тираж 77000 экз. А 05436

(Тематич. план 1976 г. № 67) Заказ 718 Цена 64 коп.

Издательство «Статистика», Москва, ул. Кирова, 39.

Московская типография № 4 Союзполиграфпрома
при Государственном комитете Совета Министров СССР
по делам издательств, полиграфии и книжной торговли,
Москва, И-41, Б. Переяславская, дом 46

64 коп.

